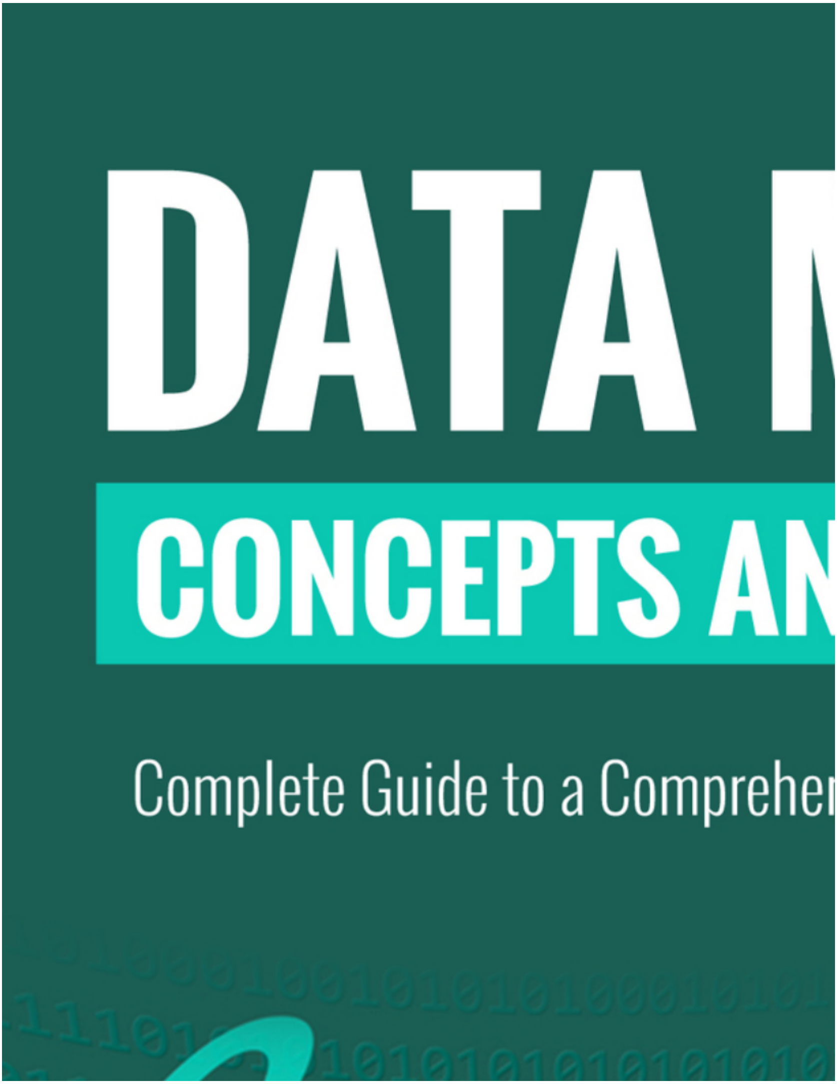

DATA M
CONCEPTS AN
Complete Guide to a Comprehen

Data Mining Concepts and Techniques

Complete Guide to a Comprehensive Understanding of Data Mining

By

Cameron Zak

© Copyright 2020 by Cameron Zac- All rights reserved.

This document is geared towards providing exact and reliable information in regards to the topic and issue covered. The publication is sold with the idea that the publisher is not required to render accounting, officially permitted, or otherwise, qualified services. If advice is necessary, legal or professional, a practiced individual in the profession should be ordered.

From a Declaration of Principles which was accepted and approved equally by a Committee of the American Bar Association and a Committee of Publishers and Associations.

In no way is it legal to reproduce, duplicate, or transmit any part of this document in either electronic means or in printed format. Recording of this publication is strictly prohibited and any storage of this document is not allowed unless with written permission from the publisher. All rights reserved.

The information provided herein is stated to be truthful and consistent, in that any liability, in terms of inattention or otherwise, by any usage or abuse of any policies, processes, or directions contained within is the solitary and utter responsibility of the recipient reader. Under no circumstances will any legal responsibility or blame be held against the publisher for any reparation, damages, or monetary loss due to the information herein, either directly or indirectly.

Respective authors own all copyrights not held by the publisher.

The information herein is offered for informational purposes solely, and is universal as so. The presentation of the information is without contract or any type of guarantee assurance.

The trademarks that are used are without any consent, and the publication of the trademark is without permission or backing by the trademark owner. All trademarks and brands within this book are for clarifying purposes only and are owned by the owners themselves, not affiliated with this document.

Table of Contents

Chapter 1: Introduction of Data Mining, How Is This Relevant?
- 1.1 Data Mining applies to What kind of data?
- 1.2 Data Mining Functionalities
- 1.3 Classification of Data Mining Systems

Chapter 2: Data Mining Processing
- 2.1 Descriptive Data Summarization
- 2.2 Data Cleaning as a Process
- 2.3 Reduction of Data
- 2.4 The Generation of Data Discretization and Conceptual Hierarchy

Chapter 3: Data Warehouse and OLAP Technology
- 3.1 What is a Data Warehouse?
- 3.2 The Operational Database Systems and Information Warehouses
- 3.3 Concept of Hierarchies
- 3.4 The Architecture of Data Warehouse
- 3.5 A three-tier Architecture for Warehouse Information
- 3.6 From Data Collection to Data Mining
- 3.7 Data Mining Distributed Systems
- 3.8 Analytics Tools Used for Data Mining

Chapter 4: Data Mining And Machine Learning
- 4.1 Machine Learning And Statistics
- 4.2 The Processing And Ethics Of Data Mining
- 4.3 Data Mining Applications
- 4.4 Frequent Patterns of Data Mining
- 4.5 FP Growth Algorithms
- 4.6 Sequential Algorithms Cover

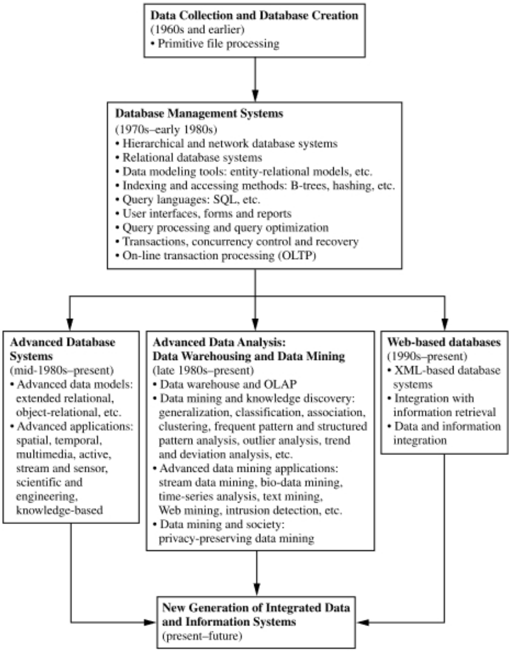
Chapter 5: Predictive Modelling of Data
- 5.1 Common Predictive Modelling Techniques
- 5.2 Regression Model Assumptions
- 5.3 Neural Networks
- 5.4 Classification and Prediction of Data
- 5.5 Approaches for Data Classification
- 5.6 Advantages and Disadvantages of Bayesian Methods

Chapter 6: Data Mining: Decision Tree Methods
- 6.1 How a Decision Tree is Grown
- 6.2 Entropy Reduction or Information Gaining

6.3 Alternate Representations for Decision Trees
Chapter 7: Artificial Neural Networks of Data Mining
7.1 Neural Data Mining Networks
7.2 Neural Network For Time Series
Chapter 8: Memory-Based Reasoning and Collaborative Data Filtering
Chapter 9: Association Rules and Market Basket Analysis
9.1 Defining An Analysis of Market Baskets
9.2 Quality of Data Analysis
Chapter 10: Link Analysis
10.1 The Theory of Basic Graphs
10.2 A Family Relation Research Framework
Chapter 11: Automatic Clusters Detection In Data Mining
11.1 K-Means Clustering
11.2 Gaussian Model
Summary

Chapter 1: Introduction of Data Mining, How Is This Relevant?

In recent years, data mining has attracted significant interest in the information industry and society as a whole, owing to the broad availability of vast quantities of data and the imminent need to turn these data into essential information and knowledge. The information and expertise gained can be used in applications ranging from market analysis, fraud detection, and consumer retention, to manufacturing control and scientific research.


Data Collection and Database Creation
(1960s and earlier)
• Primitive file processing
Database Management Systems
(1970s–early 1980s)
• Hierarchical and network database systems
• Relational database systems
• Data modeling tools: entity-relational models, etc.
• Indexing and accessing methods: B-trees, hashing, etc.
• Query languages: SQL, etc.
• User interfaces, forms and reports
• Query processing and query optimization
• Transactions, concurrency control and recovery
• On-line transaction processing (OLTP)
Advanced Database Systems
(mid-1980s–present)
• Advanced data models: extended relational, object-relational, etc.
• Advanced applications: spatial, temporal, multimedia, active, stream and sensor, scientific and engineering, knowledge-based
Advanced Data Analysis: Data Warehousing and Data Mining
(late 1980s–present)
• Data warehouse and OLAP
• Data mining and knowledge discovery: generalization, classification, association, clustering, frequent pattern and structured pattern analysis, outlier analysis, trend and deviation analysis, etc.
• Advanced data mining applications: stream data mining, bio-data mining, time-series analysis, text mining, Web mining, intrusion detection, etc.
• Data mining and society: privacy-preserving data mining
Web-based databases
(1990s–present)
• XML-based database systems
• Integration with information retrieval
• Data and information integration
New Generation of Integrated Data and Information Systems
(present–future)

The evolution of information technology or IT can be seen in data mining. In the development of the following functionalities, the database system industry has witnessed an evolutionary direction (Figure 1.): data collection and database creation, data management (including data storage and retrieval and database transaction processing), and advanced data analysis (including data storage and data mining). For example, the early development of frameworks for data collection and database construction served as a prerequisite for the subsequent development of successful mechanisms for data storage and retrieval, and for query and transaction processing. Advanced data analysis has inevitably become the next goal with various database systems providing query and transaction processing as standard practice.

Database and IT has developed from rudimentary file processing systems to sophisticated and efficient database systems since the 1960s. Since the 1970s, research and development of database systems have progressed from early hierarchical and network database systems to developing relational database systems (where data is stored of relational table structures), data processing techniques, and methods of indexing and access. Additionally, users gained easy and scalable access to data through query languages, user interfaces, streamlined processing of requests, and transaction management. Effective online transaction processing (OLTP) methods, where a query is treated as a read-only transaction, have made a major addition to the development and wide adoption of relational technology as a major tool for effective storage, retrieval, and management of large volumes of information.

Since the mid-1980s, database technology has been characterized by the widespread adoption of relational technology and an increase in research and development activities on modern and efficient database systems. These encourage the development of advanced models of data such as extended-relation, object-oriented, object-relational, and deductive models. Application-oriented database systems have flourished, including a spatial, temporal, digital, active, stream, and sensor databases, knowledge bases, and office information bases. Issues related to distribution, diversification, and data sharing have been extensively studied. Heterogeneous database systems and global information systems focused on the Internet, such as the World Wide Web (WWW), have also arisen and played an important role in the information industry.

Over the past three decades, the steady and remarkable advancement of computer hardware development has led to vast quantities of powerful and inexpensive computers, data processing equipment, and storage devices. This technology offers a great boost to the database and information industry and provides a large number of databases and information servers for transaction processing, information retrieval, and data analysis.

The data can now be stored in several different types of databases and repositories of information. One data center architecture that has arisen is the data warehouse, a center of several heterogeneous data sources clustered at a single location under a centralized scheme to promote decision making in the management. Data warehouse technology includes data integration, data cleaning, and online analytical processing (OLAP), i.e., analytical techniques with functionalities such as summary, consolidation and aggregation, and the ability to view information from different angles. While OLAP tools support multidimensional analysis and decision-taking, in-depth analysis involves additional data analysis tools, such as data classification, clustering, and interpretation of data changes over time. In addition, over and above repositories and data centers, large volumes of data can be collected. Typical examples include the data streams and the World Wide Web (www), where data flows in and out like streams, such as video surveillance, telecommunications, and sensor networks. Effective and reliable data processing in these different forms is becoming a daunting task.

The data abundance, coupled with the need for powerful tools for data analysis, has been characterized as a situation rich in data but poor in knowledge. The exponentially increasing volume of data collected and processed in vast and multiple data repositories have far surpassed our human comprehension capacity without powerful resources. As a consequence, data stored in large data repositories are "data tombs"—data archives seldom visited. Consequently, important decisions are often made not on the basis of information-rich data stored in data repositories, but on the basis of the opinion of a decision-maker, simply because the decision-maker does not have the resources to extract the valuable knowledge embedded in the vast amount of data. Additionally, find device development experts who usually depend on users or domain experts to manually insert information into a knowledge base. This method is sadly prone to biases and mistakes and is highly time-consuming and expensive. Data mining tools perform data analysis and can discover essential trends of data, significantly contributing to business strategies, knowledge bases, and science, and medical research. The widening distance between data and intelligence demands a systematic creation of data mining tools that will turn data tombs into knowledge "golden nuggets."

Simply put, data mining refers to extracting information from vast volumes of data, or "mining." Indeed, the word is a misnomer. Recall that gold mining from rocks or sand is called gold mining, rather than rock or sand mining. Thus, data mining would have been named "information mining from data" more accurately, which is, sadly, a bit long. "Data mining" does not reflect the focus on mining from large volumes of data in a shorter term. But mining is a vibrant concept that characterizes the method that finds a limited collection of precious nuggets from a lot of raw material (Figure 1.3). Thus, a common option has become such a misnomer that carries both "data" and "mining." Many other words, such as knowledge mining from data, information extraction, data/pattern analysis, data archaeology, and data dredging, have a similar or slightly different definition to data mining.

Many people view data mining as synonymous with another popularly used term, Knowledge Data Discovery, or KDD. Alternatively, some see data mining as a mere necessary phase in the information discovery process. The exploration of information as a mechanism is represented in Figure 1.4 and comprises an iterative sequence of the following steps:

1. Cleaning of data (to eliminate noise and incoherent data)
2. Integration of data (where multiple data sources can be connected together)1
3. Selection of data (where data related to the research function is retrieved from the database)
4. Transformation of data (for example, when data is transformed or consolidated into permitted mining forms by conducting summary or aggregation operations)2
5. Data mining (an integral process where smart methods for extracting data patterns are applied)
6. Evaluation of patterns (to classify the truly interesting patterns of information based on such interesting measures)
7. Presentation of knowledge (where visualization and representation of information techniques are used to present mined information to the user)

Steps 1 to 4 are specific ways of pre-processing of data, in which the data is prepared for mining. The phase-in data mining may communicate with the user or with a knowledge base. The interesting patterns are introduced to the user and can be retained in the knowledge base as new information. Notice that, according to this view, data mining is only one step in the entire process, although it is important as it uncovers hidden trends for assessment. We believe that data mining represents a phase in the discovery process of information. However, the term data mining is becoming more common in business, media, and the database analysis milieu than the longer-term extraction of information from data. Therefore, we choose to use the word data mining in this book. We take a broad view of the nature of data mining: data mining is a method of extracting useful information from vast volumes of data contained in databases, data warehouses, or other repositories.

The architecture of a typical data-mining program may have the following major components, based on this viewpoint:

Database, data warehouse, World Wide Web, or another archive of information: this is one or more databases, data centers, spreadsheets, or other forms of repositories of information. Techniques for data cleaning and data integration can be implemented on the data.

Database or data warehouse server: Depending on the user 's request for data mining, the database or data warehouse server is the solution for collecting the necessary data.

1- In the information industry, a recent trend is to conduct data cleaning and data integration as a pre-processing stage, where the resulting data is stored in a data store.

2-Data transformation and consolidation are often carried out before the data selection process, particularly in the case of data storage. Data reduction can also be made without losing its quality to achieve a smaller representation of the original data.

Knowledgebase: This is the domain information used to direct the search or determine the interestingness of the trends that arise. Such awareness may include hierarchies of concept, used to organize attributes or attribute values into various abstraction levels. Information such as user expectations can also be included and can be used to determine the interestingness of a pattern based on its unexpectedness. Other examples of domain knowledge are additional limitations or levels of interest, and metadata (e.g., the definition of data from several heterogeneous sources).

Data mining engine: This is important for the data mining system and preferably consists of a series of functional modules for tasks such as characterization, association, and analysis of associations, classification, prediction, cluster analysis, outer analysis, and analysis of evolution.

Pattern evaluation module: This part usually uses measures of interest and interacts with the modules of data mining to concentrate the quest for interesting patterns. It can use thresholds of interest to filter out trends that were discovered. Alternatively, depending on the implementation of the data mining system used, the pattern evaluation module may be combined with the mining module. It is strongly recommended to move the evaluation of pattern interestingness as far as possible into the mining process for efficient data mining so that the search is limited to only the interesting patterns.

User Interface: This module interacts between users and the data mining system, enabling users to interact with the system by defining a data mining query or mission, providing information to help guide the search, and performing exploratory data mining based on the intermediate data mining results. This portion also allows the user to search database and data warehouse schemes or data structures, analyze mined patterns, and display patterns in various ways.

From the viewpoint of data warehouses, data mining can be seen as an advanced stage of on-line analytical analysis (OLAP). However, by integrating more sophisticated methods for data analysis, data mining goes far beyond the limited reach of summary-style analytical processing in data warehouse systems.

While there are many "data mining tools" on the market, not all of them are able to conduct true data mining activities. A data analysis program that does not manage huge volumes of data should be defined more accurately as a machine learning system, a statistical data analysis method, or a prototype of an innovation system. A system that can only shape data or retrieval of information, like finding aggregate values, or performing deductive query responses in large databases should be classified more properly as a database system, information retrieval system, or deductive database system.

Data mining includes the application of multidisciplinary techniques, such as database and data warehouse technologies, analytics, machine learning, high-performance computing, pattern recognition, neural networks, data visualization, knowledge retrieval, image and signal processing, and spatial or temporal data analysis. In our description of the data mining in this book, we adopt a database perspective. That is, the emphasis is put on effective and scalable techniques for data mining. In order to be scalable, an algorithm should increase its running time approximately linearly in proportion to the size of the data, given the available machine resources such as main memory and disk space. It is possible to extract from databases and view or search from different angles by performing data mining, interesting knowledge, regularities, or high-level information. The expertise that already has been developed can be applied to decision making, process control, information management, and question handling. Consequently, data mining is considered to be one of the most critical frontiers in database and information systems and one of the most exciting interdisciplinary advances in IT.

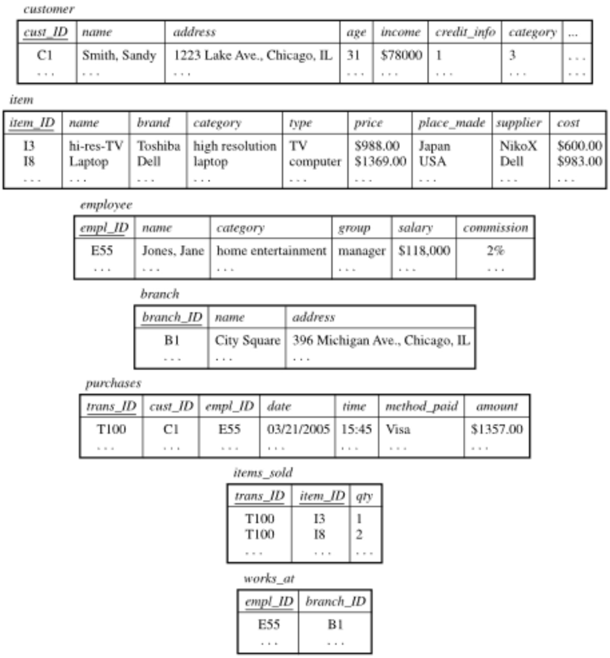
1.1 Data Mining applies to What kind of data?

In this section, we are investigating a variety of different sources of data on which mining can be carried out. Data mining should be applicable in principle to any form of data store, as well as to transient data such as data streams. The scope of our data repository review will, therefore, include relational databases, data warehouses, transactional databases, advanced database systems, flat files, data streams, and the World Wide Web. Advanced database systems include object-relational databases and specialized application-oriented databases, including spatial databases, time-series databases, text databases, and multimedia databases. For each repository system, the mining challenges and techniques can differ.

Although this book assumes readers have basic information system expertise, we include a brief introduction to each of the major data repository systems mentioned above. We also introduce the fictional AllElectronics store in this section, which will be used in the text to demonstrate concepts.

Based on Relational Data

A database system, also called DBMS, consists of a series of interrelated data, known as a database, and a set of software programs for handling and accessing the data. Software systems provide mechanisms for specifying database structures; for data storage, for simultaneous, mutual, or distributed access to data; and for maintaining the integrity and reliability of the stored information following system crashes or attempts to access it unauthorized.



customer

cust_ID	*name*	*address*	*age*	*income*	*credit_info*	*category*	...
C1	Smith, Sandy	1223 Lake Ave., Chicago, IL	31	$78000	1	3	...
...	...	...	...	...	...	...	...

item

item_ID	*name*	*brand*	*category*	*type*	*price*	*place_made*	*supplier*	*cost*
I3	hi-res-TV	Toshiba	high resolution	TV	$988.00	Japan	NikoX	$600.00
I8	Laptop	Dell	laptop	computer	$1369.00	USA	Dell	$983.00
...	...	...	...	...	...	...	...	...

employee

empl_ID	*name*	*category*	*group*	*salary*	*commission*
E55	Jones, Jane	home entertainment	manager	$118,000	2%
...	...	...	...	...	...

branch

branch_ID	*name*	*address*
B1	City Square	396 Michigan Ave., Chicago, IL
...	...	...

purchases

trans_ID	*cust_ID*	*empl_ID*	*date*	*time*	*method_paid*	*amount*
T100	C1	E55	03/21/2005	15:45	Visa	$1357.00
...	...	...	...	...	...	...

items_sold

trans_ID	*item_ID*	*qty*
T100	I3	1
T100	I8	2
...	...	...

works_at

empl_ID	*branch_ID*
E55	B1
...	...

A relational database related to set of tables, each with a specific name assigned to it. -- table is composed of a collection of attributes (columns or fields) and typically stores a wide set of tuples (records or lines). Growing tuple in a relational table is an entity defined by a unique key and represented by a set of values for the attributes. For relational databases, a semantic data model, such as an entity-relationship (ER) data model, is also constructed. The database is defined by an ER data model as a set of entities and their relations.

Include the example below:

Example -AllElectronics relational database. The company AllElectronics is defined by the following tables of relation: customer, object, employee, and branch. See Figure for fragments of the tables mentioned here.

The customer relationship consists of a collection of attributes, including a unique customer identification number (custom identification number), customer name, address, age, occupation, annual income, credit details, category, etc.

Similarly, each object, employee, and branch of relations consists of a set of attributes that define their property.

Tables may also be used to represent the relationships between or between tables with multiple relations. Which include, for example, transactions (customer purchases products, generating a sales transaction done by an employee), items get sold (lists items sold in a given transaction), and works at (employee works in an AllElectronics branch);

Database queries are written in a relational query language, such as SQL, or with the aid of graphical user interfaces, can access logical data. For example, in the above, the user may use a menu to define the attributes to be included in the application, and the constraints on those attributes. A provided question is translated into a set of

Relational data operates on entering, choosing, and projection and optimizing for optimal processing afterward. A query allows for the retrieval of specified data subsets. Suppose your task is to analyze Data from AllElectronics. Using relational queries, you can ask things like, "Show me a list of all items sold in the last quarter." Relative languages also include aggregate functions like a number, avg (average), count, max (maximum), and min (minimum) functions. These steps allow you to ask things like "show me last month's total sales, grouped by branch," or "how many sales transactions happened in December's month? "Or" Which sales individual had the most sales? When applying data mining to relational databases, we can go further by looking for trends or patterns in data.

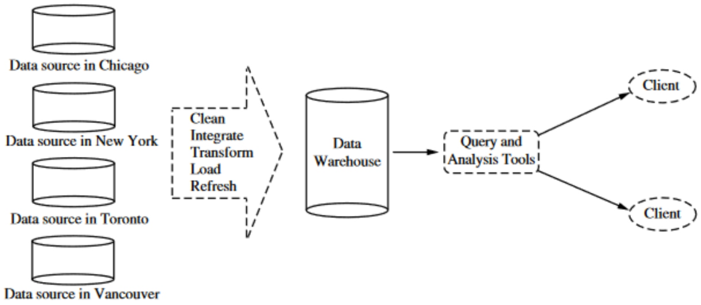
Data mining systems, for example, can analyze customer data to predict new customers ' credit risk based on their income, age, and previous credit information. In addition, data mining systems can detect anomalies, such as products whose sales are far from those predicted compared to the previous year. These anomalies can then be explored further (e.g., has there been a shift in the labeling of such products, or a significant price increase?). Relational databases are one of the most frequently available and rich sources of information and are thus a major data source in our data mining research.

Data Stocks or Warehouse

Suppose AllElectronics is an internationally successful company with branches around the world.

Every division shall have its own database collection. AllElectronics president has asked you to provide a third-quarter analysis of the company’s sales per item type by division. This is a difficult task, especially as the relevant data are distributed over multiple databases, physically located at numerous sites. If AllElectronics had a data warehouse, it would be easy to do this job.

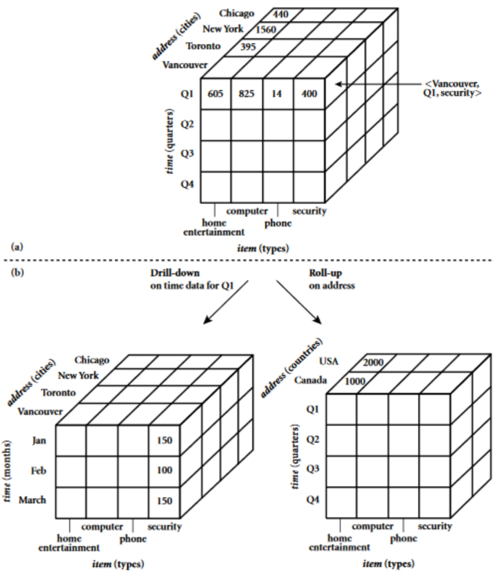
A data warehouse is actually a repository of information gathered from multiple sources, stored under a unified schema, and typically residing on a single site. Data warehouses are developed through a data cleaning process, data integration, data transformation, data loading, and regular data refreshing process. This process is dealt with in Chapters 2 and 3. Figure 1.7 illustrates the standard structure for the AllElectronics building and using a data warehouse. The data in a data warehouse is organized around major subjects, such as customer, object, manufacturer, and operation, to promote decision making.


Data source in Chicago
Data source in New York
Data source in Toronto
Data source in Vancouver
Clean
Integrate
Transform
Load
Refresh
Data Warehouse
Query and Analysis Tools
Client
Client

The data are processed and usually summarized to provide information from a historical perspective (such as from the past 5–10 years). For example, the data warehouse may store a summary of transactions per item type for every store or that summarized to a higher level, for each sales area, rather than storing the details of each sales transaction. A data warehouse is typically based on a multidimensional database structure, where each dimension corresponds to an attribute or (set) of attributes within the scheme, and each cell stores the value of some aggregate metric, such as count or sales number.

A data warehouse's actual physical structure can be a relational data store or a multidimensional data cube. A data cube offers a multidimensional view of the data and allows the pre-computation and easy access to the condensed data—example An AllElectronics data cube.

Figure, provides a data cube for a description of AllElectronics sales results. The cube has three dimensions: address (with city values in Chicago, New York, Toronto, Vancouver), time (with quarter values in Q1, Q2, Q3, Q4), and object (with object type values in home entertainment, machine, telephone, security). The aggregate value contained in each cube cell is the sum of the sales (in thousands). For example, for products related to security systems in Vancouver, the total revenue for the first quarter, Q1, is $400,000 as stored in the cell (Vancouver, Q1, defines). Additional cubes can be used to store aggregate amounts in excess of each dimension, corresponding to the aggregate values obtained using specific SQL group-bys (e.g., total sales per city and quarter, city and element, per quarter and component, or per individual dimension). "I have heard of data marts, too.



(a)
address (cities)
Chicago
New York
Toronto
Vancouver
440
1560
395
time (quarters)
Q1
Q2
Q3
Q4
605
825
14
400
<Vancouver, Q1, security>
home entertainment
computer
phone
security
item (types)
(b)
Drill-down on time data for Q1
Roll-up on address
address (cities)
Chicago
New York
Toronto
Vancouver
time (months)
Jan
Feb
March
150
100
150
home entertainment
computer
phone
security
item (types)
address (countries)
USA
Canada
2000
1000
time (quarters)
Q1
Q2
Q3
Q4
home entertainment
computer
phone
security
item (types)

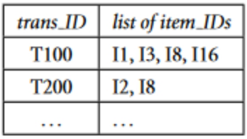
How does a data warehouse vary from a data mart? "May you know. A data warehouse collects information about issues that cover a whole company, and thus its reach is enterprise-wide. In comparison, a data mart is a subset of a data warehouse department. It focuses on selected subjects, and thus has department-wide reach. Data warehouse systems are well suited for online analytical analysis, or OLAP, by offering multidimensional data views and pre-calculation of condensed results.

OLAP operations use context information about the domain of the data being studied to allow data to be presented at different abstraction levels. These operations match different perspectives of the customer. Examples of OLAP operations include drill-down and roll-up, as shown in Figure 1.8(b), enabling the user to display data at various degrees of summation. For example, we can drill down on quarter by quarter sales data to see the data summarized by month. Similarly, to show the country-summarized data, we can roll up on sales data summarized by region. Although data warehouse tools help support data analysis, additional data mining tools are required to allow for more in-depth and automated analyzes.

Next Chapters offers an overview of the data warehouse and OLAP technology. Advanced problems-the implementation of data warehouse and OLAP and the generalization of data are discussed.

Databases on Transaction

A transactional database generally consists of a file where each record represents a trans- operation. Usually, a transaction includes a unique transaction identity number (trans ID) and a list of transaction objects (such as items purchased in a store).



trans_ID	*list of item_IDs*
T100	I1, I3, I8, I16
T200	I2, I8
...	...

The figure shows the Fragment of AllElectronics Transactional Sales Database.
The transactional database may be associated with additional tables containing other sale-related information such as the transaction date, the customer ID number, the salesperson's ID number and the branch where the sale occurred, and so on.

Example - A AllElectronics Transactional Database. You can store transactions in a row, with one record per transaction. Figure 1.9 shows a fragment of a transactional AllElectronics database. The selling table in Figure 1.9 is a nested relationship from the relational database point of view because a set of items is included in the attribute list of item IDs. Since most relational database systems do not accept nested relational structures, the transactional database is usually either stored in a flat-file in a table-like format or unfolded into a regular relationship in a format related to that of the items sold.

As an AllElectronics database analyst, you can ask, "Show me all the things Sandy Smith purchased" or "How many transactions do I3 include item number? "It can take a search of the entire transactional database to address these queries.

Suppose you want to dig deeper into the data by asking, "Which things have sold together well? "This kind of market basket data analysis will allow you to bundle item groups together as a sales maximization strategy. For example, given the awareness that printers are usually purchased along with computers, you might offer customers buying selected computers an expensive model of printers at a discount, hoping to sell more of the expensive printers. A standard data retrieval system, like the one above, cannot address queries. However, transactional data mining systems can do so by defining regular item sets, that is, sets of items that are frequently sold together. The mining of these regular transactional data patterns is explored in the next chapters.

Modern information and data structures, and specialized technologies

Relational database systems were commonly used in business applications. With the advancement of database technology, different kinds of advanced data and information systems have arisen and are undergoing growth to meet new application requirements.

The new database applications include handling spatial data (such as maps), engineering design data (such as building design, device components, or integrated circuits), hypertext and multimedia data (including text, image, video, and audio data), time-related data (such as historical records or stock exchange data), stream data (such as video surveillance and sensor data, where data floats, etc. Such systems involve powerful data structures and flexible methodds to manage complex object structures; records of variable length; semi-structured or unstructured data; text, space-time and multimedia data; and database schemes with complex structures and dynamic changes.

Advanced database systems and unique application-oriented database systems were built in response to those needs. These include object-related database systems, time and time series database systems, spatial and spatial-temporal database systems, text and multimedia database systems, heterogeneous and legacy database systems, data stream management systems, and global information systems focused on the Web.

Although these databases or information repositories need sophisticated facilities to efficiently store, retrieve, and upgrade large volumes of complex data, they also provide fertile grounds and pose many challenging issues for data mining research and implementation.

Object-Based Databases

Object-relative databases are built based on a model of object-related data. This model extends the relational model by offering a rich form of data for managing complex objects and object orientations. Since most sophisticated applications for databases need to manage complex objects and structures, object-related databases are becoming increasingly common in industry and applications.

Conceptually, the object-relational data model inherits the basic principles of object-oriented databases, in which each entity is regarded as an object in general terms. According to the example of AllElectronics, objects can be individual workers, customers, or products. A single unit encapsulates data and code relating to an entity. Every entity has the following affiliation to it:

A set of variables describing the objects that refers to the entity-relationship attributes and relational models.

A collection of messages that can be used by the object to communicate with other objects, or with the other database system.

A series of methods, with each method of holding the code for implementing a script. Upon receiving a photo(employee) message, the method returns a value in response and returns a photo of the given receiving a message—for example, the object employee form.

Objects that share a similar set of properties can be grouped into one class of entities. -- the object is an example of its class. Object classes may be organized into hierarchies of class/subclass, such that each class reflects properties common to objects in that class. For example, a class of employees may contain variables such as name, address, and date of birth. Suppose the class, sales employee, is a class subclass, worker. An object of a salesperson will inherit all variables that pertain to its employee superclass. In addition, it has all the variables directly related to being a salesperson (for example, commission). Such a feature of class inheritance facilitates the exchange of knowledge.

Techniques for the handling of complex object structures, complex data types, class
and subclass hierarchies, property inheritance, and methods and procedures need to be built for data mining in object-relational systems.

Temporary databases, databases for sequences and time series databases

A temporary database usually stores relational data, which includes attributes related to time. These attributes can include several timestamps, each of which has different semantics—a sequence database, with or without a clear notion of time, stores sequences of ordered events. Examples include sequences of consumer browsing, sources of Web clicks, and biological sequences. A time-series database stores sequences of values or events collected over repetitive time (e.g., hourly, regular, weekly) measurements. Examples include stock exchange data obtained, inventory monitoring, and observation of natural phenomena (such as wind and temperature).

Data mining techniques may be used to evaluate the characteristics of object evolution or the pattern of object modifications in the database. This knowledge can be useful when making decisions and preparing strategies. The mining of banking data, for example, may aid in the scheduling of bank tellers based on the volume of customer traffic. Stock exchange data can be collected to reveal patterns that can help you prepare investment strategies (e.g. when is the exact time to buy AllElectronics stock?). These analyzes usually allow multiple time granularities to be described. Time can be decomposed, for example, by fiscal years, academic years, or calendar years. Years can be further decomposed into months or thirds.

Spatial and Spatio-temporal databases

Spatial databases contain information relating to the room. Examples include databases on geographic (map), very large-scale integration (VLSI) or computer-aided design databases, and databases on medical and satellite images. Spatial data can be expressed in a format raster, which consists of n-dimensional bit maps or pixel maps. A 2-D satellite image, for example, can be interpreted as raster data, where each pixel records the rainfall in a given area. Maps can be depicted in vector format, where highways, bridges, houses, and lakes are depicted as unions or overlays of simple geometric structures, such as points, lines, polygons, and the partitions and networks created by these components.

Geographic databases have various uses, ranging from forestry and ecological planning to information on the location of telephone and electrical cables, pipes, and sewage networks in the public sector. Furthermore, regional databases are widely used in vehicle navigation and dispatch systems. An example of such a taxi system will store a city map with details on one-way streets, suggested routes for traveling from region X to region Y during rush hour, and the location of hospitals or restaurants and as well as each driver's current position.

"What kind of data mining on the spatial databases can be done? "May you know. Data mining can reveal patterns that define the characteristics of houses, such as a park, located near a specified spot. Other trends may describe the climatic pattern of mountainous areas at different altitudes, or describe the increasing trend of metropolitan poverty levels based on city distances from major roads. The relationship-ships among a collection of spatial objects can be examined to find out which subsets of objects are spatially auto-correlated or connected with each other. Spatial cluster analysis may classify the clusters and outliers. In addition, spatial classification can be performed to create predictive models based on the respective set of spatial object characteristics. In addition, "spatial data cubes" can be built to organize data into multidimensional structures and hierarchies that can be used to execute OLAP operations (such as drill-down and roll-up).

A spatial database is something that stores spatial objects that change over time is called a spatiotemporal database from which it is possible to extract interesting information. For review, please, we may be able to group moving object patterns and classify some oddly moving vehicles or discern a bioterrorist attack from a regular flu outbreak based on the regional spread of a disease over time.

Multimedia Databases and Text Repositories

Text repositories are databases containing object word descriptions. Typically, such word descriptions are not simple keywords but rather lengthy sentences or paragraphs, such as product details, defect or error reports, warning messages, summary reports, notes, or other documents. Text files can be extremely unstructured (such as certain World Wide Web pages). Some text databases may be very structured, i.e., semi structured (such as email messages and several HTML / XML web pages), while others may be fairly well organized (such as databases for libraries catalogues). Using the relational database systems, text databases with extremely standard structures may usually be implemented.

"What is it that data mining will reveal on text databases? "By mining text data, general and succinct explanations of text documents, keywords or content associations, and the clustering behavior of text objects can be discovered. To this end, traditional data mining methods need to be combined with knowledge recovery techniques and the creation or use of hierarchies specifically for text data as well as discipline-oriented word classification systems (such as in biochemistry, medicine, law, or economics).

Multimedia repositories store data about images, audio, and videos. They are used in applications such as image-based content retrieval, voice-mail systems, video-on-demand systems, the World Wide Web, and speech-based user interfaces that interpret spoken commands. Multimedia databases must accommodate large objects as they can allow gigabytes of storage for data objects such as images. Additionally, specialized storage and search techniques are needed. And in order to prevent picture or sound gaps and device buffer overflows, video and audio data need real-time retrieval at a steady and fixed-rate, these data are called continuous-media data.

The storage and search techniques have to be combined with traditional data mining approaches for multimedia data mining. Promising strategies include constructing multimedia data cubes, extracting multiple functions from multimedia data, and matching similarity-based patterns.

Heterogeneous databases and databases concerning legacy

A heterogeneous database is a collection of interconnected, autonomous databases on the components. The components interact to share information and to respond to queries. Objects in that component of the database that vary greatly from objects in other component databases, by making the assimilation of their semantics into the heterogeneous database.

Because of the long history of the advancement of information technology (including the implementation of various hardware and operating systems), several businesses acquire legacy databases. A legacy database is a collection of heterogeneous databases that incorporate various types of data structures, such as relational or object-oriented databases, network databases, hierarchical databases spreadsheets, interactive databases, or file systems. In a legacy database, heterogeneous databases can be linked through intra- or inter-computer networks.

It is difficult to share information through these repositories, as it would require specific rules of transformation from one representation to another, taking into account diverse semantics. Consider, for instance, the question of sharing information between various schools about student academic success. Growing school that has its own computer system and uses its own grading and curriculum system. One university may follow a quarter system, offer three courses on database systems, and assign grades from A+ to F, while another may accept a semester system, offer two courses on databases, and assign grades from 1 to 10. Precise course-to-grade transfer guidelines between the two universities are very difficult to figure out, making the sharing of knowledge difficult. Data mining techniques can provide an interesting approach to the problem of knowledge exchange by carrying out statistical data distribution and correlation analysis and converting the data into higher, more abstract, qualitative levels (such as average, decent or excellent for student grades) from which the exchange of information can then be made simpler.

Flows of data

Many applications involve producing and analyzing a new type of data, called stream data, where the data flows dynamically into and out of an observation platform (or window). These data streams have the following unique features: massive or possibly infinite length, dynamically varying, flowing in and out in a fixed order, by allowing only one or a small number of scans, and requiring fast (often real-time) response time. Typical examples of data streams include various forms of science and engineering data, time-series data, and data generated in other complex environments, such as power supply, network traffic, stock exchange, telecommunications, Web click streams, video surveillance, and weather or environment monitoring.

Since data streams are not usually stored in any form of data archive, efficient and effective management and analysis of stream data present major challenges for researchers. Many researchers are currently studying diverse issues related to the implementation of data stream management systems. A typical query model in such a system is the continuous query model, where predefined queries continuously analyze incoming streams, gather aggregate data, report the current status of data streams, and respond to their changes.

Mining data streams include the successful discovery inside stream data of general trends and dynamic changes. For example, we would like to detect intrusions of a computer network based on message flow anomaly, which can be discovered by clustering data streams, the dynamic building of stream models, or comparing the frequent current patterns with that at a certain previous time. Most stream data reside at a very low abstraction level, while analysts are often more interested in higher and multiple abstraction levels. Multilevel, multidimensional online analyzes and mining should, therefore, also be carried out on stream data.

The Big Web

The World Wide Web and centralized information services linked to it, such as Yahoo! Google, America Online, and AltaVista, offer a range of online information services worldwide where data objects are connected to enable interactive access. Users are searching for value traverse knowledge from one object through links to another. These systems provide data mining with ample opportunities and challenges. For examination purposes, understanding user access patterns will not only help improve system design (by providing efficient access between highly correlated objects), but will also lead to better marketing decisions (e.g., placing advertisements infrequently visited documents, or providing better customer/user classification and behavioral analysis). In these distributed knowledge environments, the analysis of user access patterns is called Web usage mining (or Weblog mining).

While web pages that appear to human readers to be fancy and informative, they may be highly unstructured and lack a predefined scheme, style, or pattern. Thus, it may be difficult for computers to understand and arrange the semantic sense of different web pages in a structured way for the systematic retrieval of information and for data mining. Web services that give keyword-based searches without understanding the meaning behind the web pages can only provide users with limited support. A Web search is based on a single keyword, for example, can return hundreds of Web page pointers containing the keyword, but most of the pointers will be very weakly linked to what the user needs. Often, data mining will provide additional support than Web search services here. For instance, authoritative web page analysis based on links between web pages may help rank Web pages based on their importance, impact, and subject. Automated Web page clustering and grouping help group and organize Web pages based on their contents in a multidimensional manner. Analyzing the online culture helps define secret social networks and cultures on the Online and track their evolution. Web mining is the development of Web data analysis and mining methods that are scalable and efficient. It will help us learn about transmitting Web information in general, characterizing and classifying Web pages, and uncovering Web dynamics and the interaction and other relationships between various Web pages, users, groups, and Web-based activities.

1.2 Data Mining Functionalities

We've found various types of databases and repositories of information on which data mining can be performed. Let's look at the kinds of data trends that can be used now.

The functionalities of data mining are used to determine the patterns to be found in data mining activities. Data mining activities can usually be divided into two categories: descriptive and predictive. Descriptive mining activities are characteristic of the general data resources in the database. Predictive mining tasks perform inferences to make predictions on the current data.

For certain cases, users might have no idea what kinds of patterns would be important for their data, and may, therefore, want to search in parallel for many different pattern forms. Therefore, providing a data mining system that can mine multiple types of patterns to satisfy various user expectations or applications is critical. In addition, data mining systems should be able to discover patterns at different granularities (i.e., different abstraction levels). In addition, data mining systems should allow users to specify hints to direct or focus the search for interesting patterns. Because certain patterns do not hold for all data in the database, a measure of confidence or "trustworthiness" is typically associated with each pattern that has been found.
The functionalities of data mining and the types of patterns they can find are listed below.

Characterization and Discrimination

Classes or concepts may connect the data. In the AllElectronics shop, for example, classes of products for sale include computers and printers, and consumer terms like large donors and Budget Spenders. Description of individual classes and concepts in condensed, descriptive, and yet accurate terms can be useful. These class or concept definitions are called class/concept definitions. You may derive those descriptions.

(1) Characterization of the data by summing up the data of the class under review (often referred to as Target class) generally.

(2) data discrimination, by comparing the target class with one or more comparative classes (often referred to as comparison classes), or

(3) characterization of data as well as prejudice.

Data characterization is a description of the general characteristics or characteristics of a particular data class. A database query usually collects the data corresponding to the user-specified class. For example, in order to research the characteristics of software products whose revenues in the last year have increased by 10 percent, data related to those products can be obtained by performing a SQL query.

There are many methods for successful processing and characterization of data. Next Chapters explains the basic summaries of the data based on statistical measurements and graphs. The OLAP roll-up operation based on the data cube can be used to perform user-controlled data summarization along a given dimension. Without step-by-step user interaction, an attribute-oriented induction technique can be used to perform generalization and characterisation of data.

The data characterization output can be interpreted in different ways. For e.g., pie charts, bar charts, curves, multi-dimensional data cubes, and multi-dimensional tables, like cross tables. The resulting definitions can also be described as abstract relationships or in rule form (so-called rules of character).

Example 1.4 Characterisation of the results. A data mining program should be able to generate a report summing up the characteristics of allElectronics customers who spend more than $1,000 a year. The effect may be a general consumer profile, such as being 40–50 years old, working, and having outstanding credit scores. The program would allow users to drill down on any level, such as occupation, to view such customers by their form of work.

Class discrimination is a distinction between the general characteristics of target class system objects and the general characteristics of objects from one or a number of contrasting classes. The target and contrast classes can be defined by the user, and the corresponding data objects can be retrieved via queries about the database. For example, the consumer may want to compare the general features of software products whose sales increased by 10 percent in the last year with those whose sales over the same period decreased by at least 30 percent. The techniques used to discriminate against data are similar to those used in characterizing data.

"What do the definitions of discrimination yield? "The forms of presentation of the production are similar to those for characteristic descriptions, except the descriptions of discrimination should include comparative measures which help to differentiate between the classes of the target and the classes of contrasting. Descriptions of prejudice articulated in the form of a statute are called discriminatory rules.

Example 1.5 Discrimination over data. A data mining program should allow comparison of two classes of Customers of allElectronics, such as those who frequently shop for electronic products (More than two times a month) among those who rarely shop (i.e., fewer than three times a year) for these items. The resulting definition offers a general comparative profile of customers, such as 80 percent of customers who regularly buy computer products are between 20 and 40 years old and have a university degree, while 60 percent of customers who rarely buy these items are either seniors or teens and do not have a university diploma. Drilling down on a dimension, adding new dimensions such as occupation or such as level of income, can help to find even more discriminatory features between the two groups.

Mining Frequent (Patterns) Correlations and Associations

As the name suggests, frequent patterns are patterns that frequently occur in data. There are several types of regular patterns, including the collection of objects, subsequence, and structure of substructions. Generally, a regular itemset refers to a set of items that frequently appear together in a transactional data group, such as milk and bread. A commonly occurring subsequence, such as the trend that consumers prefer to buy a Laptop first, replaced by a (digital) camera, and then a SD card, is a sequential trend (commonly). A substructure can refer to different structural structures, such as graphs, trees, or lattices, that can be combined with item sets or subsequence. When a substructure regularly occurs, it is considered a standardized pattern (frequent). Frequent mining activities contribute to the discovery of important data-based associations and correlations.

Example 1.6 Analyzing an interaction. Suppose you as AllElectronics marketing manager would like to assess which products are commonly purchased together in the same transactions. An example of such a law, taken from the transactional AllElectronics database, is Buys (X, "computer") [support = 1 percent, trust = 50 percent] Buys (X, "software")

Where X represents a client feature. 50 percent trust, or certainty, means, what if a consumer buys a computer, they have a 50 percent chance of buying software too. A 1% help means that 1% of all transactions under review showed that device and software were purchased together. This association law requires a single replicating attribute or predicate (i.e., buys). The rules of association which contain a single predicate are called single-dimensional rules of the association. The rule above can be written simply as "machine [1 percent, 50 percent]" if the predicate notation is removed.

Suppose instead that the AllElectronics relational database relating to transactions is given to us. A data mining program can consider rules of association as
Age (X, "20...29") [support = 2 percent, confidence = 60 percent] (X, "20K...29K") [support = 2 percent, confidence = 60 percent]

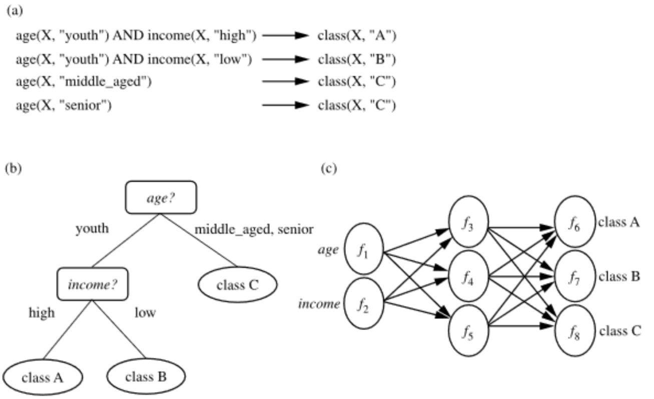
The rule suggests that 2 percent of the AllElectronics customers under study are between the ages of 20 and 29, with an income of 20,000 to 29,000 and have purchased a CD player from AllElectronics. There is a 60-65 percent chance that a consumer would purchase a CD player in this age and income category. Note that this is a relation, or predicate, between more than one attribute (i.e., age, profits, and buys). Adopting the language used in multi-dimensional databases, where each attribute is called a dimension, the above rule may be referred to as a multi-dimensional association law.

Association laws are usually dismissed as uninteresting when they meet both a minimum support threshold and a minimum confidence threshold. Additional analysis may be done to identify important statistical associations between related pairs of attribute-value.

The simplest type of frequent pattern mining is a frequent object set mining. The mining of frequent patterns, associations and correlations is discussed in Chapter 4, where the focus on efficient algorithms for frequent itemset mining is put on particular. Advanced topics are called sequential pattern mining and formal pattern mining.

Classification (Sorting) and Prediction

Classification is the process of seeking a model (or function) that defines and guides data classes or concepts in order to predict the class of objects whose class mark is unknown using the model. The derived model is mainly based on the analysis of a training data set (i.e., data objects known as the class label).


(a)
age(X, "youth") AND income(X, "high") → class(X, "A")
age(X, "youth") AND income(X, "low") → class(X, "B")
age(X, "middle_aged") → class(X, "C")
age(X, "senior") → class(X, "C")
(b)
age?
youth
middle_aged, senior
income?
class C
high
low
class A
class B
(c)
age
income
f_1
f_2
f_3
f_4
f_5
f_6
f_7
f_8
class A
class B
class C

"How does the model generated view itself? "The derived model can be represented in forms such as classification rules (IF-THEN), decision trees, mathematical formulas, or neural networks (Figure). A decision tree is a tree-like flow-chart structure in which each node denotes a test on an attribute value, each branch represents a test outcome, and tree leaves represent classes or class distributions. It is easy to convert decision trees to classification rules. A neural network is usually a series of neuron-like processing units with weighted connections between the units when used for classification. There are also other approaches for constructing classification models, such as naive Bayesian classification, supporting vector machines, and classification of k-nearest neighbours. Whereas classification predicts categorical (discrete, unordered) labels, continuous-valued functions predict models. That is, it is used to forecast numerical data values missing or inaccessible, rather than class marks. While the term prediction may refer to both numerical prediction and class mark prediction, we use it in this book primarily to refer to numerical prediction. Regression analysis is a statistical technique most commonly used for numerical prediction, though other approaches also exist.

This also includes the detection of patterns in distribution based on available data.
Classification and prediction may need to be followed by an examination of significance, which attempts to classify attributes which do not contribute to the process of classification or prediction. Then certain attributes can be ruled out.

Example 1.7 Recognition and foresight. Suppose you would like to identify a wide collection of products in the store as sales manager of AllElectronics, based on three forms of responses to a sales campaign: strong response, moderate response, and no response. From each of these three groups, you would like to derive a model based on the descriptive features of the objects, such as size, brand, place created, form, and category. The resulting classification will make a clear distinction between each class and the other, providing an ordered picture of the data set. Suppose the classification result is represented in the form of a decision tree. For example, the decision tree might define price as the single factor which best distinguishes the three groups. The tree that reveals that brand and place made include other features after price that help further differentiate artifacts of each class from another. Such a decision-making tree will help you recognize the effects of the given sales strategy and plan a more successful future strategy.

Alternatively, assume that instead of predicting categorical answer labels for each store item, you would like to predict how much revenue each item would produce during an upcoming

AllElectronics sale, based on previous sales results. It is an example of (numeric) prediction as the built model must predict a function that is continuously weighted or ordered value.

Analyzing a cluster

"Study of what cluster? "Clustering analyzes data objects without using a defined class label, unlike classification and prediction, which analyze class-labeled data objects.
In general, the class labels in the training data are not present because they are not defined, to begin with. These labels can be created using clustering. The objects are grouped or clustered based on the concept of maximizing the similarity between the intraclass and reducing similarity between the interclass. That is, clusters of objects are shaped in such a way that objects within a cluster have a strong similarity to each other but are very different from objects in other clusters. Every cluster shaped can be viewed as a class of objects from which to derive laws. Clustering may also promote the creation of taxonomy, which is to say the arrangement of findings into a hierarchy of groups grouping together related events.

Example 1.8 Review of Clusters. Cluster research on AllElectronics consumer data can be conducted to classify homogenous consumer subpopulations. Such clusters can represent marketing target groups to individual target groups. Figure 1.11 shows a 2-D customer plot with regard to customer locations in a region. There are three clusters of data points apparent.

Analysis Outliers

A database that contains data objects that don't adhere to the data's general behavior or model. Such items of data are outliers. Many methods of data mining erase.

Noise outliers or exceptions. For certain applications, however, such as fraud detection, the rare events may be more interesting than those which occur more frequently. Exterior data processing is referred to as outlier mining.

Outliers can be identified using statistical tests that follow a data distribution or probability model, or using distance measurements where items that are considered to be outliers from every other cluster. Rather than using statistical or distance measures, deviation-based approaches classify outliers by analyzing variations in a group 's principal object characteristics.

Example 1.9 Review of the Outlier. Certain analyzes that expose fraudulent use of credit cards by detecting unusually large amounts of transactions for a given account number relative to normal fees incurred by the same account. Outlier values can also be observed about the location and method of purchase or the frequency of purchase.

Analyzing Evolution

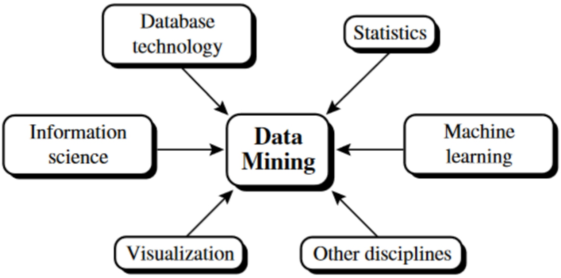
The study of data evolution identifies and models regularities or patterns for artifacts whose behavior changes over time. Although this can include characterization, discrimination, association and correlation analysis, classification, prediction, or clustering of time-related data, distinct features of such an analysis include time-series data analysis, pattern matching of sequence or periodicity, and similarity-based data analysis.

Example 1.10, Review of evolution. Suppose you have the big stock market (time-series) data available from the New York Stock Exchange for the past some years, and you would like to invest in shares of high-tech manufacturing companies. A data mining analysis of stock exchange data can recognize the regularities of stock evolution for overall stocks and stocks of particular companies. These regularities can help forecast future stock market price patterns and lead to your stock investment decisions.

1.3 Classification of Data Mining Systems

Data mining is an inter disciplinary field, the confluence of a number of disciplines, including database systems, statistics, machine learning, visualization, and science of information (Figure). In addition, techniques from other disciplines can be implemented depending on the data mining method used, such as neural networks or rough set theory, information representation, inductive logic programming, or high-performance computing. The data mining system can also incorporate techniques from spatial daily, depending on the type of data to be collected or the data mining application in question.

Data mining work is expected to produce a wide range of data mining programs due to the breadth of disciplines that contribute to data mining. Therefore, a simple classification of data mining systems must be given, which can help potential users disaggregate between these systems and recognize those that best match their needs. Data mining systems can categorize according to various requirements, as follows:



Classification by type of mined database: A data mining method may be categorized according to the form of a mined database. Database systems may be categorized according to various requirements (such as data models or data types or applications involved), each requiring its own data mining technique. Thus, data mining systems can be categorized accordingly.

For example, we can have a relational, transactional, object-relational, or data warehouse mining framework when classifying according to data models. We may have a spatial, time-series, text, stream data, interactive data mining system, or a World Wide Web mining system when classifying according to the different types of data treated.

Classification according to the types of information extracted: Data mining systems can be classified according to the types of knowledge they mine, i.e., based on data mining functionalities such as characterization, filtering, analysis of connections and relationships, classification, prediction, clustering, outer analysis and analysis of evolution. A robust data mining framework typically offers functionalities for multiple and/or integrated data mining.

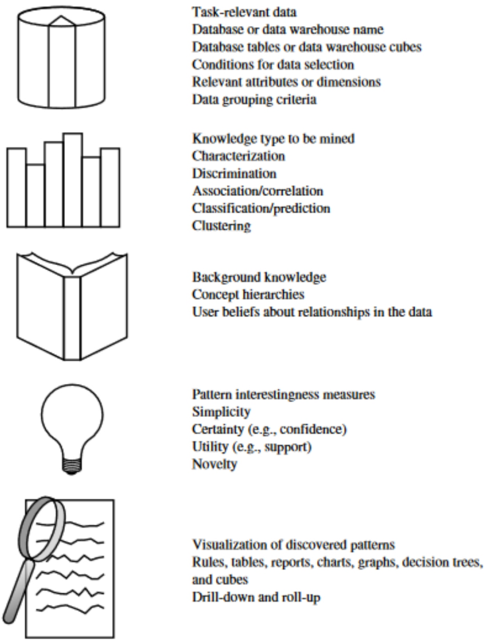
In addition, data mining systems can be differentiated based on the granularity or abstraction levels of information obtained, including generalized knowledge (at a high level of abstraction), basic knowledge (at a raw data level), or multi-level knowledge (considering several levels of abstraction). An advanced data mining program can enable information discovery at multiple abstraction levels. It is also possible to categorize data mining systems as those who mine data regularities (commonly occurring patterns) versus those who mine data anomalies (such as exceptions or outliers). In general, data regularities of mine definition overview, interaction and correlation analysis, classification, prediction, and clustering, rejecting outliers just like noise. Such techniques can help to spot outliers, too.

Classification according to the techniques used: Data mining systems can be classified according to the techniques used underlying data mining. Such techniques can be defined by the degree of user interaction involved (e.g., autonomous systems, interactive exploratory systems, query-driven systems) or data analysis approaches used (e.g., database-oriented or data warehouse-oriented techniques, machine learning, statistics, visualization, pattern recognition, neural networks, etc.). Sometimes, a sophisticated data mining method adopts several data mining techniques or works out an efficient, optimized technique that blends the benefits of a few approaches.

Classification according to adapted applications: Data mining systems can also be classified according to the applications tailored to them. Data mining tools, for example, can be applied specifically to banking, telecommunications, DNA, financial markets, e-mail, etc. Various systems also allow application-specific methods to be implemented. A standardized, all-purpose data mining program can not necessarily suit the domain-specific mining tasks.

Primitive Data Mining Function

The growing consumer should have in mind a task of data mining, that is, some sort of data analysis that he or she wishes to have performed. A data mining function may be defined as a data mining query that is inserted into the data-mining program. In terms of the primitive task of data mining, a data mining query is specified. Such primitives enable the user to interact inter-actively during exploration with the data mining system to guide the mining processor to analyze the findings from various angles or depths. The primitives for data mining define the following, as shown in Figure.


Task-relevant data
Database or data warehouse name
Database tables or data warehouse cubes
Conditions for data selection
Relevant attributes or dimensions
Data grouping criteria
Knowledge type to be mined
Characterization
Discrimination
Association/correlation
Classification/prediction
Clustering
Background knowledge
Concept hierarchies
User beliefs about relationships in the data
Pattern interestingness measures
Simplicity
Certainty (e.g., confidence)
Utility (e.g., support)
Novelty
Visualization of discovered patterns
Rules, tables, reports, charts, graphs, decision trees, and cubes
Drill-down and roll-up

The collection of task-relevant data to be extracted: This determines the sections of the database or the data set that the user is interested in. This involves fascinating database attributes or measurements of data warehouse (referred to as the appropriate attributes or measurements).

This type of data to be extracted: It describes the functions of data mining to be done, such as characterization, filtering, analysis of association or correlation, classification, prediction, clustering, outer analysis, or analysis of evolution.

The context information to be used in the discovery process: This information about the domain to be mined is useful to guide the process of knowledge discovery and to determine the patterns found. Concept hierarchies are a common type of context knowledge that allows multi-level abstraction of data to be mined. Figure 1.14 provides an example of a defined hierarchy for the age of the attribute (or dimension). Another type of context information is user beliefs about the relationships in the data.

Interesting steps and criteria for pattern evaluation: They can be used to direct the mining processor to test the patterns discovered after discovery. Different forms of information can have specific indicators of interestingness. Interesting steps for association rules include encouragement and confidence for a test. Rules the support and trust values of which are below user-specified thresholds are deemed uninteresting.

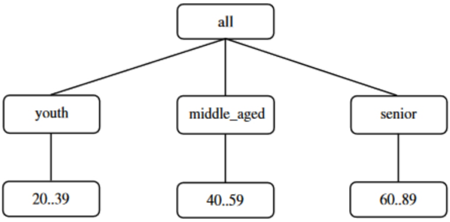
The intended representation for visualizing the patterns discovered: It refers to the manner in which the patterns discovered are to be represented, which may include rules, tables, maps, graphs, decision trees, and cubes.

A query language for data mining can be designed to integrate these primitives, enabling users to communicate flexibly with data mining systems. Using a query language for data mining offers a framework upon which to create user-friendly graphical interfaces.

It makes collaboration with other information systems and their alignment with the overall information processing environment simpler for a data-mining program.

Designing a robust data mining language is difficult because data mining encompasses a wide variety of activities, from characterizing data to analyzing the evolution. Each job has growing demands. Designing an effective query language for data mining requires a deep understanding of the strength, limitations, and underlying mechanisms of the different types of data mining tasks.

There are several proposals on languages and standards related to data mining. We use a data mining query language named DMQL (Data Mining Query Language) in this book, which was developed as a teaching tool based on the above primitives. Examples of its use to identify questions about data mining appear in this book. The language adopts a syntax similar to SQL so that it can easily be combined with the language of a relational database, SQL. Let's look at how a data mining function can be defined for it.



Reason 1.11 Laws relating to mining classification. Suppose you, as AllElectronics marketing manager, want to identify consumers based on their purchasing habits. You are particularly interested in those customers whose income is not less than $40,000 and who have purchased products worth more than $1,000, each priced at no less than

$100.00. You are especially interested in the age of the consumer, the income, the types of items purchased, the place of purchase, and where the products were produced. You want to look at the type of rules at the resulting classification. This question regarding data mining is expressed as follows in DMQL3, where each question line has been listed to assist in our discussion.

(1) use AllElectronics API database;
(2) use T. branch hierarchy place hierarchy, C.age age hierarchy
(3) Identification of the mines as good clients
(4) C.age, C. income, I. type, I. place made, T. branch relevant;
(5) from Client C, the object I, transaction T
(6) where I. item ID = C. cust ID = T. item ID = T. cust ID
And $40,000 in sales and $100 in profit
(7) T. cust-ID party
(8) Description (price) of 1,000
(9) Present as Rules

Parsing the data mining query to form a SQL query that retrieves the collection of task-relevant data defined in lines 1 and 4 to 8.

That is, line 1 specifies the database for all-electronics, line 4 lists the related attributes (i.e., on which mining is to be carried out) from the relationships listed inline 5 for the conditions specified in lines 6 and 7. Line 2 states that to generalize branch locations and customer age values, respectively, the term hierarchies location hierarchy and age hierarchy are used as context information. Line 3 specifies that classification is the sort of information to be derived for this function. Notice that we want to create a classification model for "promising customers" versus "non-promising customers." In classification, an attribute may also be defined as the class mark attribute, the values of which specifically reflect the classes. However, the two classes are implied in this case. In other words, the stated data are retrieved and considered examples of "promising customers," while the remaining of the customers in customer table are considered to be "non-promising customers." Classification is almost carried out on the base of this training package. Line 9 defines a set of rules for showing the mining results.

There is no any standard data mining query language yet, but strong progress has been made by researchers and industry in this direction. Microsoft's OLE DB for Data Mining (described in this book's appendix) includes DMX, a data mining language designed with XML. Other attempts for standardization include PMML (Programming Data Model Mark-up Language) and CRISP-DM (Cross-Industry Standard Data Mining Process).

Integration of a database or data storage system into a data mining program

The good system architecture will allow the data mining system to make the best use of the software setting, perform data mining tasks effectively and in a timely manner, interoperate and share information with other information systems, adapt to the diverse needs of users, and evolve over time.

How to combine or pair the DM system with a database (DB) system and/or a data warehouse (DW) system is a key problem in the design of a data mining (DM) system. Whether a DM system operates as a stand-alone system or is integrated into an application program, otherwise there are no DB or DW systems to deal with. This simple scheme is called no coupling, where the DM design's main focus is on designing powerful and efficient algorithms to leverage the available data sets. However, when a DM system operates in an environment requiring it to communicate with other components of the information system, such as DB and DW systems, potential integration schemes include no coupling, loose coupling, semi-tight coupling, and tight coupling. We look at each of those schemes as follows:

No coupling: No coupling means no feature of a DB or DW system would be used by a DM system. It can collect data from a given source (such as a file system), process data using some data mining algorithms, and then store the mining results in another file.

Such a system suffers from many drawbacks, albeit basic. Second, a DB framework offers more versatility and consistency in data storage, management, access and processing. A DM system can spend a considerable amount of time locating, storing, cleaning, and transforming data without using a DB / DW system. Data appears to be well structured, indexed, cleaned, incorporated, or unified in DB and/or DW systems, making it easy to find the task-relevant, high-quality data. Second, in DB and DW systems, there are many proven, scalable algorithms and data structures to implement. Efficient, scalable implementations using these systems can be realized. In addition, the bulk of data is or would be stored in DB / DW systems. A DM system would need to use other methods to extract data without any coupling of these systems, making it impossible to incorporate such a system into an information-processing environment. So, no coupling is a bad feature.

Loose coupling: IT means that a DM system uses certain DB or DW system facilities, gathers data from a data repository maintained by these systems, conducts data mining, and then stores the mining results either in a file or at a specified location in a database or data warehouse.

Loose coupling is better than no coupling because by using query processing, indexing, and other device facilities, it can get any portion of data stored in databases or data warehouses. It brings some benefits from the versatility, performance, and other functionality that these systems provide. But several loosely coupled mining systems are based on the main memory. Since mining does not explore the data structures and query optimization methods offered by DB or DW systems, high scalability, and good performance with large data sets are difficult to achieve for loose coupling.

Semi tight coupling: Semi tight coupling means that in addition to linking a DM system to a DB / DW system, the DB / DW system may provide successful implementation of a few basic data mining primitives (identified by the review of frequently found data mining functions). These primitives can include sorting, indexing, aggregating, analyzing histograms, multiway joining, and pre-calculating other important statistical measurements, such as number, count, max, min, standard deviation, etc. In addition, some frequently used intermediate mining results can be precomputed and stored in the DB / DW program. Because these intermediate mining results are either precomputed or can be efficiently computed, this design would boost the performance of a DM system.

Tight coupling: Tight coupling implies smooth integration of a DM system into DB / DW system. The sub-system for data mining is viewed as one functional part of an information system. Data mining queries and functions are designed based on an overview of the mining application, data structures, indexing systems, and DB or DW framework database processing methods. With more technical developments, DM, DB, and DW systems will develop and converge together with various functionalities as one information system. This will have a consistent atmosphere in which information is processed.

This approach is highly beneficial because it promotes the effective implementation of data mining features, high system efficiency, and an integrated ecosystem for information processing.

It is easy to see with this analysis that a Data Mining system should be combined with a DB / DW system. Loose coupling is better than no coupling, but not effective, since it uses both data and system facilities of a DB / DW system. Tight coupling is highly desirable but is non-trivial in its implementation, and further work is needed in this field. Semi tight coupling is a loose to tight coupling compromise. Identifying commonly used primitives in data mining and providing successful implementation of these primitives in DB or DW systems is important.

Chapter 2: Data Mining Processing

Imagine you're a manager at AllElectronics and were tasked with analyzing the company's sales data at your branch. You set out to accomplish this role straight away. You inspect the database and data warehouse of the company carefully, identifying and selecting the dimensions or attributes to be included in your analysis, such as item, price, and units sold. Ah, Alas! You note that there is no reported value of some of the attributes for various tuples. You would like to provide useful details for your review as to whether and the item purchased was advertised on sale, but you find that these details were not registered. In addition, users of your database system have documented bugs, odd values, and discrepancies in the recorded data for some transactions. The data you wish to analyze using data mining techniques is incomplete (lacking attribute values or certain attributes of interest, or containing only aggregate data), noisy (containing errors or outlier values that deviate from what is expected), and inconsistent (e.g., containing discrepancies in the department codes used to categorize items). Welcome to the true world!

Incomplete, noisy, and unreliable data are typical properties of massive databases and data centers in the real world. Incomplete data for a number of rea-sons can occur. Interest attributes, such as customer information regarding sales transaction data, may not always be available. Other data might not be included; simply because at the time of entry, it was not considered relevant. Relevant data cannot be reported due to confusion, or due to defective equipment. It could have removed data that were incompatible with other reported data. In fact, his- tory recording or changes to the data may have been ignored. Missing data may need to be inferred, especially for tuples with missing values for some attributes.

There are several potential explanations for the noisy data (with wrong values for the attributes). The instruments used to collect data can be inaccurate. Human or device errors may have occurred upon data entry. Errors may also occur in data transmission. There may be limitations in the technology, such as a small buffer size to manage coordinated data transmission and consumption. Incorrect or Irrelevant data may also result from inconsistencies in the naming of used conventions or data codes, or inconsistent input field formats, such as date. Duplicate tuples need cleaning of details, too.

Routines for data cleaning function on "cleaning" data by filling out missing values, smooth-input noisy data, finding or eliminating outliers, and resolving inconsistencies. If users think the data is dirty, they are unlikely to trust the results of any data mining applied to it. In addition, dirty data can cause uncertainty for procedure mining, which results in unreliable performance. While most mining routines have some procedures to handle incomplete or noisy data, they are not robust at all. Instead, they may focus on avoiding overfitting of the data to the modeled function. Hence, running your data through some data cleaning routines is a useful pre-processing phase. Section 2.3 deals with ways to clean up your results.

Going back to your mission at AllElectronics, suppose you'd like to use multi-source data in your study. It will require multiple integrations.

Databases, cubes of data, or archives, in other words, data integration. But in various databases, certain attributes reflecting a given term that have different names, creating inconsistencies and redundancies. For example, customer identification attributes may be named customer ids in one data store and custom ids in another. Inconsistencies in naming can also occur for values of the attributes. For example, in one database, the same first name may be documented as "Bill," but in another, "William," and in the third, "B." In addition, you assume those characteristics can be derived from others (e.g., annual revenue). Having large quantities of redundant data can delay or confuse the process of discovering information. It is almost clear that steps need to be taken in addition to data cleaning to help prevent redundancies during data integration. Usually, when preparing the data for a data warehouse, data cleaning, and data integration is done as a pre-processing step. In order to detect and remove redundancies that may have resulted from data integration, additional data cleaning may be carried out.

Returning to your data, you have agreed, say, that you would like to use a distance-based mining algorithm for your research, such as neural networks, neighboring classifiers or clustering Such approaches have better results if the data to be analyzed has been standardized, i.e., scaled to a limited range such as [0.0, 1.0]. For example, customer information includes the age and annual compensation attributes. The attribute of annual salary usually takes much greater value than age. Consequently, if the characteristics are left unnormalized, the distance measurements taken on the annual salary will usually outweigh the measurements taken on age. In addition, it would be helpful to provide quantitative information about the revenue per customer area for your analysis — something that is not part of any pre-computed data cube in your data warehouse. You will soon realize that operations of data transformation, such as standardization and aggregation, are additional pre-processing procedures for data that would contribute to the mining process success. Section 2.4 addresses data integration and data transformation.

"Hmmm," you think when you take the data even more into consideration. "The collection of data I have chosen for analysis is Massive, which will certainly slow down the mining process. Is there some way I can raise the size of my data collection, without losing the results of data mining? "Data reduction achieves a reduced representation of the data set, which is much smaller in volume but produces the same (or nearly the same) analytical results. There are many Data Management Techniques. These include data aggregation (e.g., the formation of a data cube), collection of sub-set attributes (e.g., elimination of irrelevant attributes by correlation analysis), reduction of dimensionality (e.g., encoding schemes such as minimum length encoding or wavelets), and reduction of numerosity (e.g. "replacement" of data by alternative, smaller representations such as clusters or parametric models). The subject of Section 2.5 is Data Reduction. Data may also be "reduced" through generalization through the use of concept hierarchies, where low-level concepts, such as city for customer position, are replaced through higher-level concepts such as area or province or state. A hierarchy of concepts organizes the concepts into different layers of abstraction. Data discretization is a method of data reduction that is very useful for the automatic generation from numerical data of concept hierarchies. That, along with the automatic generation of concept hierarchies for categorical data, as defined in Section 2.6.

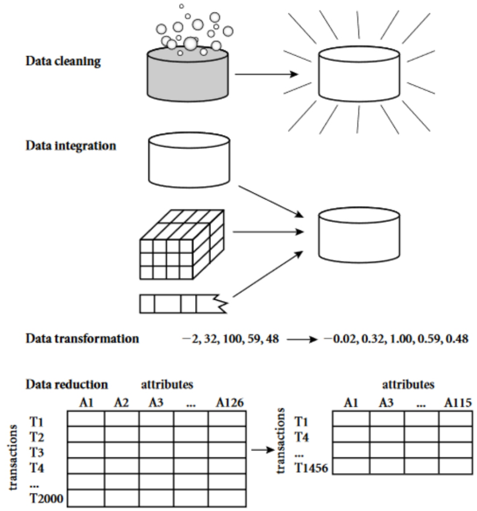

Data cleaning
Data integration
Data transformation
−2, 32, 100, 59, 48
−0.02, 0.32, 1.00, 0.59, 0.48
Data reduction
attributes
A1 A2 A3 ... A126
transactions
T1
T2
T3
T4
...
T2000
attributes
A1 A3 ... A115
transactions
T1
T4
...
T1456

Figure summarizes the steps described here for the pre-processing of the data. Notice that the above categorization does not exclude each other. The removal of redundant data, for example, can be seen as a form of data cleaning, as well as data reduction.

In short, data in the real world appears to be dirty, incomplete, and inconsistent. Data pre-processing techniques can improve data quality, thereby improving the accuracy and reliability of the subsequent mining process. Pre-processing of data is an important step in the process of knowledge discovery because quality decisions must be based on quality data. The identification of data irregularities, early rectification, and reduction of the data to be analyzed will lead to tremendous payoffs for decision making.

2.1 Descriptive Data Summarization

It is important to have an overall image of your data if data pre-processing is to be efficient. It is possible to use concise data description techniques to define the typical properties of your data and to highlight which data values should be viewed as noise or outliers. Therefore, we introduce the basic principles of concise data summarization before moving through the practical operations of pre-processing techniques of data.

Users would like to know about data characteristics for both core tendency and data dispersion for many data pre-processing tasks. Central tendency measurements include mean, median, mode, and middle, while data dispersion measurements include quartiles, interquartile range (IQR), and variance. These (descriptive) statistics are of help in interpreting the data distribution. Such measures were extensively studied in the statistical literature. Through the viewpoint of data mining, we need to look at how they can be measured effectively in large databases. Particularly the notions of distributive measure, algebraic measure, and holistic measure should be added. Understanding what sort of measure, we're dealing with will help us pick an appropriate implementation for it.

In this section, we examine at various ways of calculating the data's core tendency. The most popular and accurate numerical indicator of a data set's "center" is
Mean (arithmetic) Let it be x1, x2, xN, for example, is a set of N values or observations.
Any feature, including remuneration. This set of values is the mean of

$$\bar{x} = \frac{\sum_{i=1}^{N} x_i}{N} = \frac{x_1 + x_2 + \cdots + x_N}{N}. \qquad (2.1)$$

This is the average (avg) (in SQL) built-in aggregate function provided in relational database systems.

A distributive measure is a measure (i.e., function) that can be measured for a given set of data by partitioning the data into smaller subsets, measuring the measure for each subset, and then linking the results to arrive at the value of the measure for the original (full) set of data. Both sum) (and count) (are distributive measures, since they can be calculated this way. Others include max) (and min). An algebraic measure is a calculation that can be measured on one or more distributive measures by applying an algebraic function. Average (or mean)) (is, therefore, an algebraic measure, since it can be calculated by sum()/count). Usually, data cubes[2] , sum() and count() are saved in precalculation while computing. And it's easy to calculate the average for data cubes.

Often a weight wi every is associated with increasing value xi in a set, for I = 1,...N.
The weights represent the importance, meaning, or frequency of occurrence attached to their respective values. In this case, we can calculate the

$$\bar{x} = \frac{\sum_{i=1}^{N} w_i x_i}{\sum_{i=1}^{N} w_i} = \frac{w_1x_1 + w_2x_2 + \cdots + w_Nx_N}{w_1 + w_2 + \cdots + w_N}. \tag{2.2}$$

This is called the arithmetic weighted mean, or weighted average. Remember that one more example of an algebraic metric is the weighted average.

Although the mean is the single most useful quantity to characterize a data set, it's not always the best way to calculate the data center. A big concern with the mean is their vulnerability to extreme (e.g., outer) values. Also, the norm can be skewed by a few unreasonable values. Of example, a company's mean pay may be significantly driven up by that of a couple of high paying executives. Likewise, a couple of quite low scores could pull down the average score of a class in an exam quite a bit. Then we can use the trimmed mean to counteract the effect caused by a small number of extreme values, which is the mean produced after the high and low extremes have been cut off. For example, before calculating the mean, we can filter the observed salary values and eliminate the top and bottom 2 percent. We should avoid trimming too big a portion at both ends (such as 20 percent) because this can point to the loss of useful knowledge.

With distorted (asymmetric) data, the mean is a better representation of the data center. Suppose a given set of separate values for N is ordered in numerical order. If N is odd, then the middle value of the ordered set is the median; otherwise (i.e., if N is even), the median is the sum of the two middle values.

A holistic calculation is a calculation that has to be measured on the entire collection of data as a whole. This cannot be determined by partitioning the given data into subsets and combining in each subset the values obtained for the calculation. The median is an example of a calculation of holistic. Holistic interventions are much more costly to quantify than distributive interventions like the ones mentioned above.

But We can easily approximate a data set's median value. Assume that data is grouped according to their xi data values in intervals and that the frequency (i.e., number of data values) of each interval is known. For example, people may be grouped in intervals such as 10–20 K, 20–30 K, etc., according to their annual salary. Let the mean interval be the interval that contains the median frequency. By interpolation, we can approximate the median of the entire data set (e.g., median salary) using the formula:

$$median = L_1 + \left(\frac{N/2 - (\sum freq)_l}{freq_{median}}\right) width, \qquad (2.3)$$

The whole set of data (well f req) l is the sum of the frequencies of all the lower intervals where L1 is the lower median interval boundary, N is the median interval number, f req median is the median interval frequency, and distance.

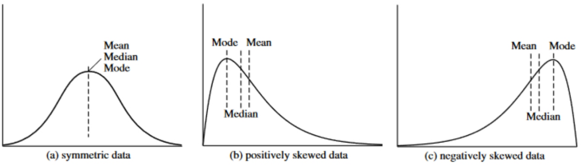


Figure 2.2 Mean, median, and mode of symmetric versus positively and negatively skewed data.

The key indicator of inclination is the mode. The most commonly occurring value in
the system is the mode for a collection of data. The greatest frequency will lead to several different values, resulting in more than one mode. Data sets are called unimodal, bimodal, and trimodal, with one, two, or three modes, respectively. A data

collection containing two or more modes is usually multimodal. On the other end, if each data value only exists once, then mode does not exist.

With moderately distorted (asymmetric) unimodal frequency curves, we have the following empirical relation:

Middle − mode = 3 selection (mean − median). (2.4) This means the unimodal frequency curve mode, which is moderately distorted, could be accurately measured if you know the mean and median values.

The mean, median, and mode all these are at the center value in a unimodal frequency curve with perfect symmetric data distribution, as shown in Figure 2.2(a). Nevertheless, the data are not symmetric in most practical applications. Alternatively, they can be either positively skewed, where the mode occurs at a value smaller than the mean (Figure 2.2(b)), or negatively skewed, where the mode occurs at a value larger than the median (Figure 2.2(c)).

Even the midrange may be used to determine a data set's central tendency. This is the sum of the set's greatest and least values. This algebraic formula can be easily calculated using the max) (and min) functions of the SQL aggregate.

2.2 Data Cleaning as a Process

Missing values, noise, and discrepancies contribute to inaccurate information. We have been looking at methods so far to tackle missing data and to smooth out results. "But clean data that's a huge challenge. How about a process for cleaning the data? Where exactly does one go about tackling this task? Are there any tools to help? The first step in cleaning data as a process is the detection of discrepancies.

Discrepancies can be caused by multiple factors, including poorly designed data entry forms with multiple optional fields, human error data entry, deliberate errors (e.g. respondents who do not wish to disclose information about themselves), and data decline (e.g. outdated address). Discrepancies can also result from inconsistent representations of data and the incoherent use of codes. Another source of inconsistencies is Errors in instrumentation devices that record data and System errors. Errors may also occur when the data is used (insufficiently) for purposes other than originally intended. Data integration may also lead to inconsistencies (e.g., where a given attribute can have different names in different databases).4 "So, how can we proceed with the detection of discrepancies? "Use any information you might already have about data properties as a starting point. This data or information is referred to as metadata, or "files." For examples, what is the domain and type of data for each attribute? Which values are acceptable for each attribute? Which is the value length range? Will all values suit within the range expected? Are there any known attribute dependencies? The descriptive summaries of data presented in Section 2.2 are useful for understanding data trends and identifying anomalies here. For example, values for a given attribute that are more than two standard deviations away from the mean may be flagged as possible outliers. You can write your own scripts in this phase and/or use some of the tools we'll discuss further below. Despite this, the noise, the outliers, and uncommon values which need analysis.

As a data analyst, you will check for the incoherent use of codes and any contradictory representations of data (such as '2004/12/25' and '25/12/2004' for the date).

Field overloading is another source of errors usually occurring when developers cram new attribute definitions into unused (bit) portions of already specified attributes (e.g., using an unused bit of an attribute whose value range only uses, say, 31 out of 32 bits). Additionally, data should be analyzed with regard to particular laws, consecutive laws and null rules.

A unique rule says that for that attribute every value of the given attribute has to be different from all other values. A sequential rule states that between the lowest and highest values for the attribute there can be no missing values and that all values must also be identical (e.g., as in test numbers). A null rule describes the use of blanks, question marks, special characters, or other strings that may signify the null condition (e.g., when a value is not available for a specified attribute), and how such values should be treated. As mentioned in Section 2.3.1, it may include reasons for missing values (1) the person originally requested to provide an attribute value refuses and/or finds that the requested information does not apply (e.g., a license number attribute left blank by non-drivers); (2) the data entry of that person does not know the correct value; or (3) the value is to be provided at a later stage of the process.

The null rule would specify about how to record the null condition, e.g. to store zero for the numerical attributes, a blank for character attributes, or any other conventions that may be in use (such as "don't know" or?" "entries should be turned blank). There are several different commercial tools available that can help in detecting discrepancy.

Simple domain knowledge (e.g. knowledge of any postal addresses and spell checking) is used by data scrubbing tools to detect errors and make data corrections. These tools rely on parsing techniques and fuzzy matching when cleaning data from multiple sources. Methods for data auditing recognize inconsistencies by reviewing the data to uncover laws and relationships, and identifying data that breaches certain conditions. They are variants of tools for the data mining. We can use statistical analysis, for example, to find similarities, or clustering to classify outliers. They can also use summaries of the descriptive data as defined in Section 2.2. Many errors in the data can be manually checked using external references.

Errors made at data entry, for example, can be resolved by doing a paper trace. However, most errors may involve the transformation of the data. This is the second stage in the process of cleaning the data. That is, if we notice inconsistencies, to fix them, we usually need to describe and apply (a sequence of) transformations. Commercial tools can help in the step of transforming data.

Data migration tools allow you to specify simple transformations, such as replacing the "gender" string with "sex" ETL (Extraction / Transformation / Loading) tools that allow users to specify transforms via a GUI. Usually these tools only support a limited set of transforms so that we can often also choose to write a custom script for this step of the data cleaning process. Iterates the two-step cycle of the identification of anomalies and the transformation of data (to fix discrepancies).

However, this process is prone to error and time consuming. Some transformations may cause further differences. Many nested inconsistencies can only be observed after the fixation of others. For example, a typo such as "20004" can only surface in a year field once all the data values have been converted to a uniform format. Transformations are often performed as a batch process, while the user is waiting without any feedback. Only after completing the transformation can the user go back and verify that no new anomalies were generated by mistake. Numerous iterations are typically necessary before the user is satisfied. Any tuples that cannot be handled by a given transformation is usually written to a file without any clarification of the reasons behind its failure. As a result, there is also lack of some interactivity to the whole data cleaning process. New data cleaning approaches underscore increased interactivity. For example, Potter's Wheel is a freely accessible data cleaning tool (see http:/control.cs.berkeley.edu / abc), which incorporates the identification and transformation of discrepancies. Gradually, users create a series of transformations by writing and debugging individual transformations on a spreadsheet-like interface, one step at a time. You may define the transformations graphically, or by giving examples. Results are displayed on the documents and are visible on the screen immediately. The user can opt to reverse the transformations so that transformations that produced additional errors can be "erased." The tool automatically tests inconsistency on the current transformed view of the data in the context. When inconsistencies are discovered, users may increasingly create and refine transformations, leading to more effective and efficient data cleaning. Another solution to improved data cleaning interactivity is the creation of declarative languages for data transformation operators to be precise.

Such research focuses on defining powerful SQL extensions and algorithms which allow users to effectively express specifications for data cleaning. As we find out more about the data, it is important that the metadata continue to be updated to reflect this knowledge.

It would help improve the cleaning of data on future versions of the same data store.

2.3 Reduction of Data

Imagine having selected data to be analyzed from the AllElectronics data warehouse. The collection of data is expected to be enormous! Complex data processing and mining will take a long time on small quantities of data, making such processing impractical or infeasible.

Data reduction methods may be implemented in order to achieve a reduced representation of the data set, which is much smaller in length, but which preserves the quality of the original data similar. That is, mining should be a bit more effective on the reduced data collection but yield the same analytical results.

Data Reduction Techniques include:

1. Aggregation of the data cube, where aggregation operations are applied to the data while building a data cube.

2. Selection of subset attributes where irrelevant, weakly relevant, or redundant attributes or dimensions may be detected and deleted.

3. Reduction of dimensionality, where encoding methods are used to reduce the scale of the data collection.

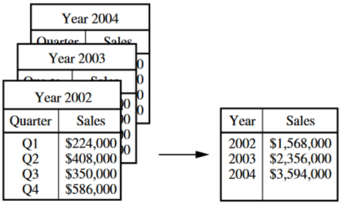
4. Reduction in numerosity, where data is replaced or estimated by alternative, smaller data representations such as parametric models (which only need to store the model parameters instead of the actual data) or non-parametric approaches such as clustering, sampling, and histogram use.

5. Generation of discretization and concept hierarchy, in which raw data values are replaced by ranges or higher conceptual levels for attributes. Data discretization is a form of a reduction in numerosity, which is very useful in the automatic generation of concept hierarchies. Discretization and concept of generation hierarchy are the powerful tools for data mining, allowing data mining at multiple abstraction levels. Therefore, we postpone the discussion of discretization and the generation of concept hierarchy to Section 2.6, which is entirely devoted to this subject.

The rest of this section discusses Strategies 1 to 4 above. The computing time spent on reducing data should not outweigh or "erase" the time saved by mining on a reduced size collection of data.

Aggregation of data in Cube

Imagine having collected the data for your research. These data comprise the quarterly sales of AllElectronics for the years 2002 to 2004. Nevertheless, you are interested in the annual revenue (total per year), rather than the quarterly average. It helps the data to be aggregated such that the resulting data represents the cumulative revenue per year rather than per quarter. The Aggregation is shown in Figure 2.13. The resulting data set is smaller in volume, without loss of information required for the task of analysis.



Year 2004

Year 2003

Year 2002

Quarter	Sales
Q1	$224,000
Q2	$408,000
Q3	$350,000
Q4	$586,000

Year	Sales
2002	$1,568,000
2003	$2,356,000
2004	$3,594,000

Figure 2.13 Sales data for a given branch of *AllElectronics* for the years 2002 to 2004. On the left, the sales

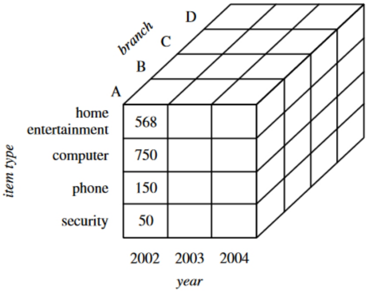


Figure 2.14 A data cube for sales at *AllElectronics*.

Data cubes are discussed in detail in Data Warehousing Chapter 3. Here we briefly introduce certain concepts. Information cubes store aggregated knowledge of a multi-dimensional nature. For example, for each branch of AllElectronics, Figure 2.14 shows a data cube for multi-dimensional analysis of sales data as regards annual sales per item type. Each cell holds an aggregate data value in multi-dimensional space corresponding to the data point. (Only certain cell values are shown for readability.) Concept level abstraction. For instance, a branching hierarchy may require branches to be grouped into regions, based on their address. Data cubes provide quick access to pre-computed, summarized data, benefiting both on-line analytical processing and data mining.

The cube which is generated at the lowest level of abstraction is called the base cuboid. The base cuboid should correspond to an individual interest entity, for example, sales or customers. In other words, the lowest level will be useful for analysis or useful. The apex cuboid is a cube to the highest degree of abstraction. The apex cuboid would give one total for the sales data in Figure 2.14—the total sales for all three years, for all types of items, and for all branches. Information cubes generated to varying degrees of abstraction are also referred to as cuboids, so that then an information cube may refer to a cuboid lattice. Each greater abstraction level further reduces the resulting data size. The smallest available cuboid relevant for the given task should be used when responding to data mining requests. Chapter 3 also tackles this issue.

Selection of Subset attributes

The data sets for review that include hundreds of attributes, many of which may be obsolete or redundant to the mining mission. e.g., if the task is to classify customers as to whether or not they are likely to buy a popular new CD from AllElectronics when notified of a sale, attributes like the telephone number of the customer are likely to be irrelevant, unlike attributes like age or music taste. Although it somehow possible for a domain expert to pick out some of the useful attributes, this may be a complex and time-consuming task, especially when the data's behavior is not well known (therefore, a reason behind its analysis!). Leaving out relevant attributes or holding irrelevant attributes may be harmful to the mining algorithm employed, creating uncertainty. It can lead to weak quality trends discovered. The added volume of unnecessary or redundant attributes will further slowdown the mining cycle.

Attribute subset selection reduces the data set size by deleting obsolete or redundant attributes (or dimensions). The aim of attribute subset selection is to get a min set of attributes so that the resulting probability of distribution data classes is as similar as possible to the original distribution obtained using all attributes. An extra benefit is mining on a reduced set of attributes. This reduces the number of attributes found in the patterns discovered, helping to promote comprehension of the patterns.

"What are the 'good' subsets of the original attributes? "There are 2n possible subsets for n attributes. Search for the exhaustive optimal subset of any attributes can be prohibitively high, particularly as n increases and the number of data classes. Therefore, heuristic methods which may explore a reduced search space are commonly used for attribute subset selection. In this, these methods are typically greedy while searching through Space attributes, they always make what seems the best option at the time. The strategy is to make a locally optimal choice will lead to a solution that is optimal globally. In practice, these greedy methods are efficient and may come close to estimating an optimal solution.

Usually, the "best" (and "worst") attributes are calculated using statistical significance tests, which assume the attributes are independent of each other. Many other measurements for assessing attributes can be used, such as the information gain measure used for classification decision trees in the building.

Specific heuristic methods for selecting a subset of attributes include the following strategies, some of which are shown in Figure 2.15.

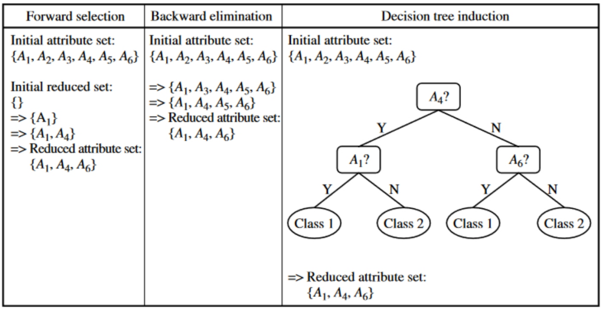

Forward selection
Backward elimination
Decision tree induction
Initial attribute set:
$\{A_1, A_2, A_3, A_4, A_5, A_6\}$
Initial reduced set:
{}
=> $\{A_1\}$
=> $\{A_1, A_4\}$
=> Reduced attribute set:
$\{A_1, A_4, A_6\}$
Initial attribute set:
$\{A_1, A_2, A_3, A_4, A_5, A_6\}$
=> $\{A_1, A_3, A_4, A_5, A_6\}$
=> $\{A_1, A_4, A_5, A_6\}$
=> Reduced attribute set:
$\{A_1, A_4, A_6\}$
Initial attribute set:
$\{A_1, A_2, A_3, A_4, A_5, A_6\}$
A_4?
Y
N
A_1?
A_6?
Y
N
Y
N
Class 1
Class 2
Class 1
Class 2
=> Reduced attribute set:
$\{A_1, A_4, A_6\}$

Figure 2.15 Greedy (heuristic) methods for attribute subset selection.

1. Stepwise forward selection: The process begins as a reduced collection with an empty set of attributes. Determines the best of the original attributes and adds them to the reduced set. The best of the original attributes remaining is added to the collection at each subsequent iteration or step.

2. Stepwise backward elimination: The entire set of attributes begins the process. The worst attribute remaining in the set is removed at every level.

3. Combination of forwarding selection and reverse elimination: step-by-step selection and reverse elimination methods can be combined to pick the best attribute at each step and eliminate the worst from the remaining attributes.

4. Decision tree induction: Algorithms for the decision tree, such as ID3, C4.5, and CART, was initially intended for classification. Decision tree induction constructs a flowchart — like structure in which each internal (non-leaf) node denotes an attribute test, each branch can corresponds to a test result, and each external (leaf) node denotes a class prediction. The algorithm selects the "best" attribute for partitioning the data into single groups at every node.

By using decision tree induction for sub-set selection of attributes, a tree is built from the given data. All attributes that do not appear in three I considered insignificant. The set of attributes which appear in the tree makes up the reduced subset of attributes.

The criteria for stopping the methods may vary. The procedure can use a thresh-old on the measure used to decide when to stop the selection process for the attribute.

Reduced Dimensionality

Data encoding or transformations are implemented in dimension reduction in order to achieve a simplified or "compressed" representation of the original data. When the original data can be retrieved without any loss of information from the compressed data, then data reduction is considered lossless. When then we can only reconstruct an approximation of the original data, then the reduction of the data is considered lossy. String compression has several well-tuned algorithms. While usually lossless, they allow data to be manipulated only limitedly. Instead, we focus in this section on two popular and effective methods of reduction of loss dimensionality: wavelet transforms and analysis of main components.

2.4 The Generation of Data Discretization and Conceptual Hierarchy

Data discretization techniques may be employed by splitting the range of the attribute into intervals to reduce the quantity of values for a given continuous attribute. Then you can use interval marks to replace the real data values. Replacing various values of a continuous variable with a small number of interval labels thus eliminates the original data and simplifies this. This leads to a concise, simple to use, representation of mining results at the knowledge-level.

Discretization methods can be classified depending on how discretization is done, such as how it uses class knowledge or direction (i.e. top-down vs. bottom-up) that it continues. When the method of discretization uses class knowledge then we assume discretization is supervised. Otherwise it is unregulated. Whether the cycle starts by first selecting one or a few points (called breakpoints or cut points) to break the entire set of attributes, and then repeats this recursively at the resulting intervals, it is called top-down discretization or split. It compares with bottom-up discretization or merging, starting with the evaluation of all continuous values as possible split-points, eliminating some by combining neighborhood values into intervals, and then applying it method recursively to the resulting intervals. Discretization can be performed recursively on an attribute to provide a hierarchical or multi-resolution partitioning of the values of the attributes, known as hierarchy of definition. Concept hierarchies are useful for various levels of abstraction in mining.

A definition hierarchy describes discretization of the attribute for a given numerical attribute. By gathering and replacing low-level concepts (such as numerical values for attribute age) with higher-level concepts (such as youth, middle-aged, or senior), concept hierarchies may be used to minimize data. While information is lost by the genetic realization of these data, the simplified data may be more relevant and interpretable. This contributes to a consistent representation of the results of data mining among multiple mining tasks, a common requirement. Additionally, mining on a reduced collection of data requires less input / output operations, which is more effective than mining on a larger set of extremely large data. Due to these advantages, discretization techniques and concept hierarchies are usually implemented as a pre-processing step before data mining, rather than during mining. An example of a hierarchy concept for the attribute price is shown in Figure For the same attribute, more than one definition hierarchy can be specified to meet the needs of different users.

It can be time-consuming and tedious task for a user or a domain expert to manually define concept hierarchies. Fortunately, different discretization methods can be used to generate concept hierarchies for numerical attributes automatically or dynamically refine those. In addition, several hierarchies for categorical attributes within the database schema are implicit and can be described automatically at the specification level of the scheme.

Let us look at how concept hierarchies are generated for numerical and categorical data.

The Generation of Discretization and Conceptual Hierarchy for Number Data

Due to the large variety of potential data ranges and the regular changes of data values, it is difficult and laborious to define definition hierarchies for numerical attributes. These manual specifications can also be somewhat arbitrary.

Concept hierarchies for numerical attributes can be automatically built based on discretization of the data. They analyze the following methods: binning, evaluating histograms, discretization dependent on entropy, cluster analysis, and intuitive partitioning discretization. Typically speaking, each approach assumes the values to be discrete are ordered in ascending order.

Binning: It is a method of top-down separation, based on a given number of bins. Section 2.3.2 discussed Data Smoothing binning methods. These methods are also used as methods of discretization for reducing numerosity and generating hierarchies of concepts. For example, attribute values can be discrete by applying a binning of equal width or equal frequency, and then replacing each bin value by the mean or median bin, as in bin media smoothing or bin medians smoothing, respectively. To generate ate definition hierarchies these techniques can be applied recursively to the resulting partitions. Binning does not use class information, and therefore is an unattended technique of discretization. This is sensitive to the number of bins defined by the user, and to the existence of outliers.

Analysis of historygrams

Like binning, they are analyzing histogram is an unsupervised technique of discretization, since it does not use class information. Histograms partition the values for an attribute called buckets, A, into disjoint ranges. Section 2.2.3 introduced historygrams. Section 2.5.4 defined the partitioning rules for describing histograms. For example, in a histogram of equal width, the values are partitioned into partitions or ranges of equal size (such as in price in Figure 2.19, where each bucket has a width of $10). The values are partitioned with a histogram of equal frequency, so that ideally every partition contains the same number of tuples of data. To order to automatically create a multilevel concept hierarchy, the histogram analysis algorithm can be applied recurrently to each partition, with the process terminating until a predetermined number of concept levels has been achieved. The recursive procedure can also be regulated with a minimum interval size per point. It defines the minimum partition distance, or the minimum number of values at each point, for each partition. Histograms can also be partitioned, as described below, based on clustered data distribution analysis.

Discretization Based on Entropy

Entropy is one of the discretization steps most widely employed. Through seminal work on knowledge theory and the idea of obtaining information, Claude Shannon first proposed it. Entropy-based discretization is a method of supervised separation from the top down. In its calculation and determination of the split-points (data values for partitioning an attribute range), it explores class distribution information. For discrete a numerical variable, A, the method selects the value of A that has the lowest entropy as a split-point, and the resulting intervals are partitioned recursively for arriving at a hierarchical discretion. Such discretization shapes a hierarchy of definitions to A.

Let D consist of a set of attributes and a class-label attribute defined by the data tuples.
The class-label attribute provides information about the class per tuple. The basic method for discretization of an attribute A within the set based on entropy is as follows:

1. Value of each A can be considered as a potential interval boundary or split-point to partition the A range. In other words, a split point for A can split the tuples in D into two subsets that meet the conditions A and A.
A > split point, produces a binary discretization, respectively.
2. As mentioned above, Entropy-based discretization uses information regarding the

Tuple Grade Mark. We will take a look at classification to illustrate the theory behind entropy-based discretization. Suppose we want to divide the tuples into D by splitting them into attribute A and some split-point. Ideally, we would like to see this partitioning proceed to an exact tuple classification. For instance, if we had two classes, we would hope all tuples of, say, Class C1 would fall into one partition, and all tuples of Class C2 would fall into the other. This, however, is doubtful. For example, the first partition may contain a lot of C1 tuples but also some C2 tuples. How much information would we still need after this partitioning, for a perfect classification? The expected information requirement for classifying a tuple in D based on partitioning by A is called this number.

Analyzing clusters

Cluster analysis is a common method of discretization of data. A clustering algorithm can be implemented by partitioning the values of A into clusters or classes to discrete a numerical attribute, A. Clustering takes into account both the distribution of A and the closeness of data points and is thus capable of generating discretization results of high quality. Clustering concept can be used to generate a concept hierarchy for A by either following a top-down splitting strategy or a bottom-up merging strategy, in which each cluster forms a node of concept hierarchy. Each initial cluster or partition may be decomposed into several subclusters in the former, forming a lower hierarchy level. In the latter, clusters are formed by combining adjacent clusters repeatedly to form concepts of a higher level. Chapter 11 explores clustering methods for data mining.

Chapter 3: Data Warehouse and OLAP Technology

Throughout multidimensional space, the data warehouses generalize and organize data. Information warehouse construction includes data cleaning, data integration, and data transformation, which can be seen as an important pre-processing stage for data mining. In addition, data warehouses provide online analytical processing (OLAP) tools for interactive analysis of varied granularity multidimensional data that facilitate successful data generalization and data mining. Many other functions of data mining, such as grouping, classification, prediction, and clustering, can be combined with OLAP operations to boost immersive information mining at multiple levels of abstraction. The data warehouse has also become an increasingly important tool for data collection and electronic analytical processing, which can provide an efficient data mining tool.

Hence data storage and OLAP are important steps in the process of information discovery. This chapter provides a summary of the OLAP and data center technologies. Such an outline is important for understanding the entire process of data mining and information discovery. In this chapter, we review a well-accepted data warehouse concept to see why more and more companies are constructing data warehouses to analyze their results. The data cube, which is a multidimensional data model for the data warehouses and OLAP, as well as OLAP operations such as roll-up, drill-down, slicing, and dicing, was studied in particular. We also look at the architecture of the data warehouse, including measures on the design and development of the data warehouse. A description of the implementation of data warehouses discusses genetic methods for effective data cube storage, OLAP data indexing, and OLAP query processing. Lastly, we look at online computational mining, a versatile framework combining data warehouse and OLAP technology with data mining technology.

3.1 What is a Data Warehouse?

Data warehousing provides business managers with architectures and tools to coordinate, interpret, and use their data routinely to make strategic decisions. In today's dynamic, fast-growing environment, data warehouse systems are important resources. Many business people have invested millions of dollars in the last couple of years constructing enterprise-wide data warehouses. Many people feel that data warehousing is the latest must-have marketing weapon with competition mounting in every industry – a way to retain customers by learning more about their needs.

"What is a data warehouse then, exactly? "In many ways, data warehouses have been defined, which makes it difficult to formulate a precise definition. A data warehouse which refers to a database that is kept separately from the operational databases of an organization. Data warehouse systems allow an array of application systems to be integrated. They further support information processing by providing a solid platform for the analysis of consolidated historical data.

According to William H. Inmon, a pioneering architect in data warehouse systems design, "A data warehouse is a subject-oriented, distributed, time-varying and non-volatile data set of support of the decision-making phase of management" [Inm96]. This short but detailed description describes the important characteristics of a data warehouse. The four keywords, subject-oriented, standardized, time-variant, and non-volatile distinguish data warehouses from other data storage structures, like relational database systems, transaction processing systems, and file systems. Let's look at each of those main features more closely.

Subject-oriented: A data warehouse is structured around major topics such as customer, manufacturer, commodity, and sales. Instead of focusing on an organization's everyday activities and transaction management, a data warehouse focuses on modeling and analyzing data for decision-makers. Therefore, data warehouses usually offer a clear and succinct view of particular topic issues by removing data that is not useful in the decision support process.

Integrated: Typically, a data center is designed by combining several heterogeneous sources such as relational databases, flat files, and transaction records online. Techniques of data cleaning and data integration are implemented to ensure continuity in naming conventions, encoding systems, measurements of attributes, and so on.

Time-variant: Data are processed from a historical perspective (e.g., the past 5–10 years) to provide context. Each main structure inside the data warehouse includes an element of time, either indirectly or explicitly.

Non-volatile: A data center is often a physically separate data store created from the data contained in the operating environment. Because of this separation, a data warehouse does not need processes for processing transactions, recovery, and competition control. In data accessing, it normally needs only two operations: initial data loading and data access.

In summary, a data warehouse is like a semantically compatible data store that acts as a physical implementation of a data model that facilitates decision making and holds the knowledge an organization needs to make strategic decisions on. Also, a data warehouse is also seen as an infrastructure built by the incorporation of data from several heterogeneous sources to enable organized and/or ad hoc searches, analytical reporting, and decision-making.

Based on this data information, we view data warehousing as the process of building data warehouses and using them. Creating a data center includes data maintenance, data integration, and data consolidation. The use of a data center also requires a range of technologies to facilitate decision making. This allows to "knowledge workers" (e.g., managers, analysts, and executives) to use the warehouse to quickly and conveniently gain an overview of the data, and to make sound warehouse-based decisions. Many writers use the term "data warehousing" to refer only to the data warehouse construction process, while the term "DBMS warehouse" is used to refer to data warehouse management and use. We are not going to make the distinction here.

"How do the organizations use data warehouse information? "This information or data is used by many companies to facilitate business decision-making, including
(1) The consumer attention, which involves the study of customer purchase patterns (such as purchasing habits, purchasing time,

budget cycles and spending appetites);
(2) repositioning and controlling product portfolios by evaluating market output by quarter, year and geographic regions in order to fine-tune production strategies; (3) reviewing and looking operations;

Even from the point of view of heterogeneous database integration, data warehousing is very useful. Usually, many organizations gather different kinds of data and retain vast databases from numerous, heterogeneous, autonomous, and dispersed sources of knowledge. It is highly desirable but difficult to incorporate these data and provide easy and efficient access. A lot of work load was put into achieving this aim in the database industry and the research community.

In addition to multiple, heterogeneous databases, the conventional database approach to heterogeneous database integration is to create wrappers and integrators (or mediators). A metadata dictionary is used when a query is posted to a client site to convert the query into queries that are suitable for the individual heterogeneous sites involved. Then these queries are mapped and sent to local processors for the query. Returned responses from the different sites are combined into a global response package. This query-driven approach involves complex processes of filtering and incorporation of information and competes with processing at local sources for resources. For frequent queries, it is inefficient and potentially costly, especially for queries that require aggregations.

Data warehousing provides a fascinating alternative to the conventional heterogeneous database integration method mentioned above. Alternatively, of using a query-driven approach, data warehousing uses an update-driven approach in which various, heterogeneous sources of information are combined in advance and processed in a warehouse for direct querying and analysis, unlike databases that process transactions electronically, data warehouses do not hold the most recent information. A data center, however, adds high efficiency to the distributed heterogeneous database system as data is copied, pre-processed, merged, annotated, compiled, and restructured into one semantic data store. In fact, the handling of queries in data centers does not conflict with the handling at local sources. In addition, data warehouses can store and integrate historical information and support complex multidimensional queries. This has made data warehousing popular in the industry.

3.2 The Operational Database Systems and Information Warehouses

Since most people are familiar with commercial relational database systems, comparing these two types of systems makes it easy to understand what a data warehouse is.

Digital operating database systems have the main role in conducting on-line trans-action and query processing. These systems are called applications for on-line transaction processing (OLTP). These cover much of an organization's day-to-day activities, such as buying, production, fabrication, insurance, payroll, registration, and accounting. On the other hand, data warehouse systems represent consumers or technical workers in the data collection and decision-making positions. These systems are capable of organizing and displaying data in varied formats to meet the various user needs. Such applications are known as electronic analytical processing systems (OLAP).

The key characteristics which distinguish OLTP from OLAP are summarized as follows:

Users and system orientation: An OLTP program are customer-oriented and is used by clerks, customers, and information technology professionals to manage transactions and requests. An OLAP program is market-oriented and is used by experienced staff, including administrators, executives, and analysts for data processing.

Data content: An OLTP framework handles current data, which is usually too complex to be easily used in decision making. An OLAP framework collects vast volumes of historical data, provides a description and sorting facilities, and stores and maintains information at varying granularity levels. Such features ease the use of the data in informed decision making.

Database Design: An OLTP framework usually adopts data model entity-relationship (ER) and application-oriented database architecture. Typically, an OLAP program adopts either a snowflake or a star model (to be addressed in Section 3.2.2) and a subject-oriented database architecture.

View: An OLTP program focuses solely on the existing data within a business or agency, without regard to historical data or data in different organizations. In contrast, due to an organization's evolutionary process, an OLAP system often spans multiple versions of a database schema. OLAP systems also manage the information coming from multiple entities, incorporating information from several data stores. OLAP data are stored on various storage media because of their large volume.

Access patterns: The OLTP system's access patterns consist primarily of short, atomic transactions. Such a framework needs frameworks for competition management and recovery. Access to OLAP systems, however, is mostly read-only operations (because most data warehouses store historical information rather than up-to-date information), while there may be complex queries.

Certain characteristics that differentiate between OLTP and OLAP systems include database size, operating frequency, and performance metrics. These are summed up in Table 3.1.

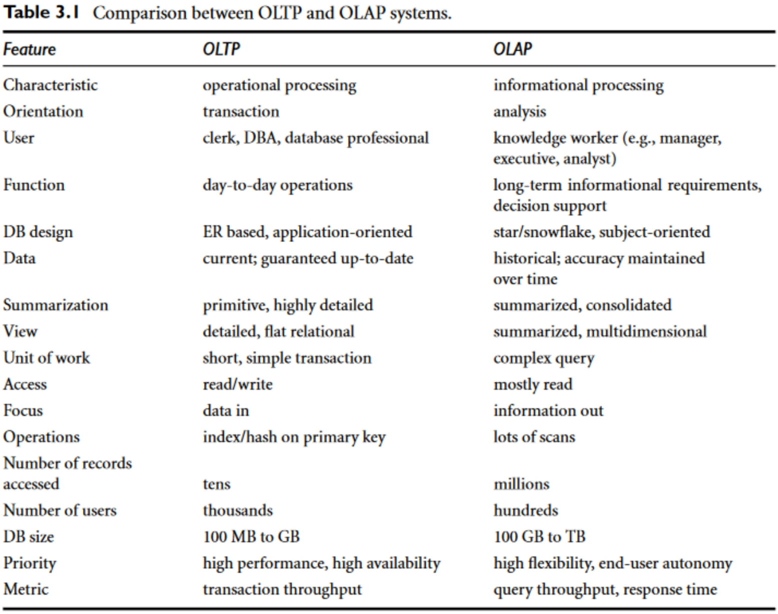


Table 3.1 Comparison between OLTP and OLAP systems.

Feature	*OLTP*	*OLAP*
Characteristic	operational processing	informational processing
Orientation	transaction	analysis
User	clerk, DBA, database professional	knowledge worker (e.g., manager, executive, analyst)
Function	day-to-day operations	long-term informational requirements, decision support
DB design	ER based, application-oriented	star/snowflake, subject-oriented
Data	current; guaranteed up-to-date	historical; accuracy maintained over time
Summarization	primitive, highly detailed	summarized, consolidated
View	detailed, flat relational	summarized, multidimensional
Unit of work	short, simple transaction	complex query
Access	read/write	mostly read
Focus	data in	information out
Operations	index/hash on primary key	lots of scans
Number of records accessed	tens	millions
Number of users	thousands	hundreds
DB size	100 MB to GB	100 GB to TB
Priority	high performance, high availability	high flexibility, end-user autonomy
Metric	transaction throughput	query throughput, response time

So, why do you have a data warehouse separately?

Since operational databases store enormous quantities of data, you might ask, "Why not conduct on-line analytical processing directly on these databases instead of investing additional time and money to create a separate data warehouse? "Helping foster high efficiency of both systems is a big explanation for such a split. An operational database is designed and optimized from known tasks and workloads, such as indexing and hashing with primary keys, searching for specific records, and optimizing "canned" queries. On the other hand, questions relating to data warehouses are always complex. They include computing large data groups at condensed levels and may require the use of multi-dimensional view-based methods of special data management, access, and implementation. The handling of OLAP queries in operational databases will degrade the efficiency of operational tasks considerably.

In fact, an operating database facilitates several transactions being handled at the same time. Mechanisms for concurrent control and recovery, such as locking and logging, are necessary to ensure transaction continuity and robustness. For summarization and aggregation, an OLAP application also needs read-only access to data records. Competitiveness control and recovery mechanisms that jeopardize the execution of concurrent transactions if implemented for these OLAP operations and thus significantly reduce the OLTP system throughput.

Finally, the separation of operating databases from data warehouses is based on the different data structures, contents, and uses of both systems. The support decision includes historical data, while operational databases usually do not hold historical data. In this sense, however, abundant the data for operational repositories is typically far from full for decision-making. Support for decision-making allows data from heterogeneous sources to be combined (such as aggregation and summary), resulting in high-quality, clean, and integrated data. Operational databases, on the other hand, contain only comprehensive raw data such as transactions that need to be compiled prior to review. Since the two systems provide very different functionalities and require different data types, separate databases are currently required. Most operational relational database management systems vendors, however, are beginning to develop these systems to accommodate OLAP queries. If this pattern continues, it is predicted that the distinction between OLTP and OLAP systems will decrease.

A Multi-dimensional Model of Data

A multi-dimensional data model is based on data warehouses and OLAP software. This model views the data in a data cube format. You'll learn how the data cubes model n-dimensional data in this section. You will also learn about theory and concept hierarchies and how they can be used to enable interactive mining at multiple levels of abstraction in simple OLAP operations.

From Tables and Sheets to Cubes of Data

"What a cube for data? "A data cube enables multi-dimensional visualization and display of the data. Its dimensions and facts define it.

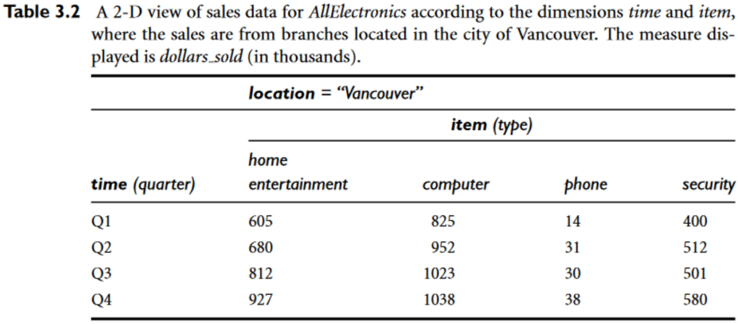


Table 3.2 A 2-D view of sales data for *AllElectronics* according to the dimensions *time* and *item*, where the sales are from branches located in the city of Vancouver. The measure displayed is *dollars_sold* (in thousands).

	location *= "Vancouver"*			
	item *(type)*			
time *(quarter)*	*home entertainment*	*computer*	*phone*	*security*
Q1	605	825	14	400
Q2	680	952	31	512
Q3	812	1023	30	501
Q4	927	1038	38	580

Dimensions are, in general terms, the experiences or individuals that an entity wants to keep information about. AllElectronics, for example, can create a sales data warehouse to keep records of the store's sales regarding the time, object, division, and location measurements. Such measurements allow the store to keep track of things like monthly item sales and the branches and locations where the items were being sold. Every dimension will have a related table, called a dimension table, which defines the dimension further. For example, an item dimension table can contain the name, brand, and type of item attributes. Users or experts may define the dimensional tables or automatically generate and change based on the distribution of data.

Typically, a multi-dimensional data model is organized around a central theme, like, for example, sales. A fact table shows this trend. Numerical measures are true. Quantities by which we want to examine inter-dimensional interactions. Examples of facts for a sales of data warehouse include sold dollars (sales in dollars), sold units (number of units sold), and budgeted amounts. The table of facts includes the names of the data, or steps, as well as the keys to each of the relevant tables. When we look at multi-dimensional schemas, you'll soon get a better picture of how this works.

Although we usually think or consider of cubes as 3-D geometric constructs, the data cube is n-dimensional in data warehousing. To gain a better understanding and knowledge of data cubes and the multi-dimensional data model, let's begin by looking at a simple 2-D data cube that is actually a table or spreadsheet for AllElectronics sales data. We will look in particular at the AllElectronics sales data for products sold in the city of Vancouver every quarter. Table 3.2 displays those results. The sales for Vancouver are displayed in this 2-D representation with respect to the time dimension (organized in quarters) and the object dimension (organized according to the types of products sold). The displayed fact or measure is sold in dollars (in thousands).

Table 3.3 A 3-D view of sales data for *AllElectronics*, according to the dimensions *time*, *item*, and *location*. The measure displayed is *dollars_sold* (in thousands).

	location = *"Chicago"*				***location*** = *"New York"*				***location*** = *"Toronto"*				***location*** = *"Vancouver"*			
	item				***item***				***item***				***item***			
time	*home ent.*	*comp.*	*phone*	*sec.*	*home ent.*	*comp.*	*phone*	*sec.*	*home ent.*	*comp.*	*phone*	*sec.*	*home ent.*	*comp.*	*phone*	*sec.*
Q1	854	882	89	623	1087	968	38	872	818	746	43	591	605	825	14	400
Q2	943	890	64	698	1130	1024	41	925	894	769	52	682	680	952	31	512
Q3	1032	924	59	789	1034	1048	45	1002	940	795	58	728	812	1023	30	501
Q4	1129	992	63	870	1142	1091	54	984	978	864	59	784	927	1038	38	580

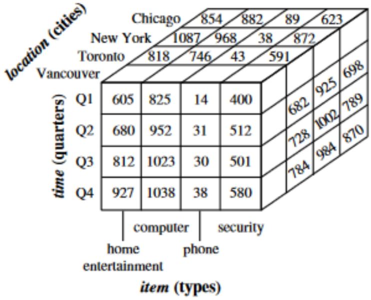

location (cities)
Chicago
854
882
89
623
New York
1087
968
38
872
Toronto
818
746
43
591
Vancouver
time (quarters)
Q1
605
825
14
400
Q2
680
952
31
512
Q3
812
1023
30
501
Q4
927
1038
38
580
682
925
698
728
1002
789
784
984
870
computer
security
home entertainment
phone
item (types)

Figure 3.1 A 3-D data cube representation of the data in Table 3.3, according to the dimensions *time*, *item*, and *location*. The measure displayed is *dollars_sold* (in thousands).

Now, suppose we want the sales data to be viewed with a third dimension. For example, suppose if we want to display the data by time and object, as well as a location for the cities of Chicago, New York, Toronto, and Vancouver. Table 3.3 displays certain 3-D details. Table 3.3 3-D data is represented as a set of 2-D tables. In visual terms, as in Figure 3.1, we can also represent the same data in the form of a 3-D data cube.

3.3 Concept of Hierarchies

A concept hierarchy describes a mapping sequence from a collection of low-level concepts to more general, higher-level concepts.

Consider a hierarchy of concepts for the location of the dimension. Location City values include Vancouver, Toronto, New York, and Chicago. But any city can be mapped to the province or state it belongs to. For instance, it can map Vancouver to British Columbia, and Chicago to Illinois. In addition, the provinces and states that be compared to the nation they belong to, such as Canada or the USA. These mappings form a concept hierarchy for dimension position, mapping a collection of low-level (i.e., cities) concepts to higher-level, more general (i.e., countries) concepts. Figure 3.7 shows the definition Hierarchy mentioned above.

Within the database scheme, many concept hierarchies are implicit. For example, suppose that the position of the dimension is defined by the number, street, city, attributes.

A hierarchy of definitions such as "street < town < province or state < nation" This Province or State hierarchy, zip code, and region. These attributes are associated with a total order, as shown in Figure 3.8(a).

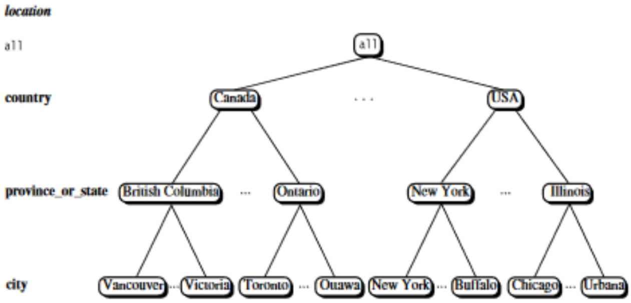

location
all
all
country
Canada
USA
province_or_state
British Columbia
Ontario
New York
Illinois
city
Vancouver
Victoria
Toronto
Ottawa
New York
Buffalo
Chicago
Urbana

Figure 3.7 A concept hierarchy for the dimension *location*. Due to space limitations, not all of the nodes of the hierarchy are shown (as indicated by the use of "ellipsis" between nodes).

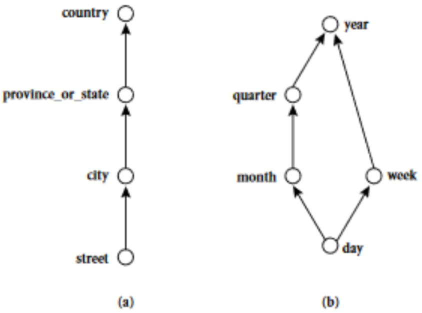

country
province_or_state
city
street
(a)
year
quarter
month
week
day
(b)

Figure 3.8 Hierarchical and lattice structures of attributes in warehouse dimensions: (a) a hierarchy for *location*; (b) a lattice for *time*.

This is the absolute or partial order of the attributes in a database scheme is called the hierarchy schema. Concept hierarchies common to many applications, such as the concept hierarchy for time, can be pre-fined in the data mining framework. Data mining systems should provide versatility for users to customize predefined hierarchies to their specific needs. For example, users may want to identify a fiscal year beginning on April 1 or an academic year beginning on September 1.

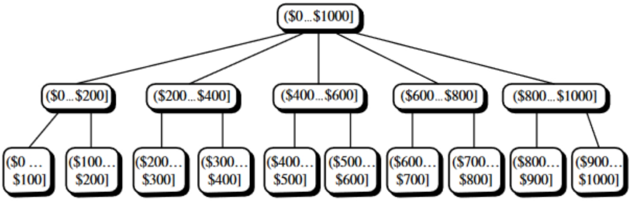


Figure 3.9 A concept hierarchy for the attribute *price*.

OLAP Multi-dimensional data model operations

"Why are the hierarchies of definitions useful at OLAP? "Data are organized into multiple dimensions in the multi-dimensional model, and each dimension includes several abstraction levels identified by definition hierarchies. The company offers users the ability to display data from various perspectives. There is a variety of OLAP data cube operations to materialize these various views, enabling the data at hand to be interactively queried and analyzed. OLAP thus provides a user-friendly environment for collaborative analysis of the results.

OLAP processes, Example 3.8. For multi-dimensional data, let's look at some typical OLAP operations. Figure 3.10 shows any of the operations mentioned below. A data cube for AllElectronics sales lies at the core of the figure. The cube contains the position, time, and item dimensions, where the position is aggregated with respect to city values, time is aggregated with regard to quarters, and object is aggregated with respect to item types. We refer to this cube as the main cube, to aid in our clarification. The displayed measure is sold in dollars (in thousands). (Only certain cell values of the cubes are displayed for better readability.) The data analyzed are for the cities of Chicago, New York, Toronto, and Vancouver.

Roll-up: The roll-up process (also called by some vendors the drill-up process) performs aggregation on a data cube, either by moving up a design hierarchy for a dimension or by reduction of dimensions. Figure 3.10 shows the outcome of a roll-up operation carried out on the central cube by climbing up the position hierarchy of the definition given in Figure 3.7. This hierarchy was described as the "street of the complete order."

< City < province or state < country. "Aggregates were shown during the roll-up operation Data by increasing the hierarchy of the area from city level to city level
State-Country. In other words, the resulting cube groups the data by region, rather than grouping the data by area.

If the dimension reduction is made by a roll-up, one or more dimensions are extracted from the given cube. Consider, for example, a sales data cube that contains only the position and time of the two dimensions. Roll-up can be achieved by eliminating, say, the time dimension, leading to an aggregate of overall revenue by region, rather than by region and by time.

Drill-down: The opposite of roll-up is the Drill-down. It navigates to more detailed data from less detailed data. Through either stepping down a design hierarchy for one dimension or adding additional dimensions, drill-down can be realized. Figure 3.10 demonstrates the outcome of a drill-down procedure carried out on the central cube by walking a Time hierarchy defined as "day < month < trimester < year."

Happens by moving down the time ladder from the quarter stage to the more comprehensive Monthly Standard. The resulting data cube reports the actual monthly revenue rather than a quarterly overview of those.

Because a drill-down adds more extra detail to the data given, it can be done by adding new dimensions to a cube, too. For example, by introducing an additional dimension, such as a customer group, a drill-down can occur on the central cube of Figure 3.10.

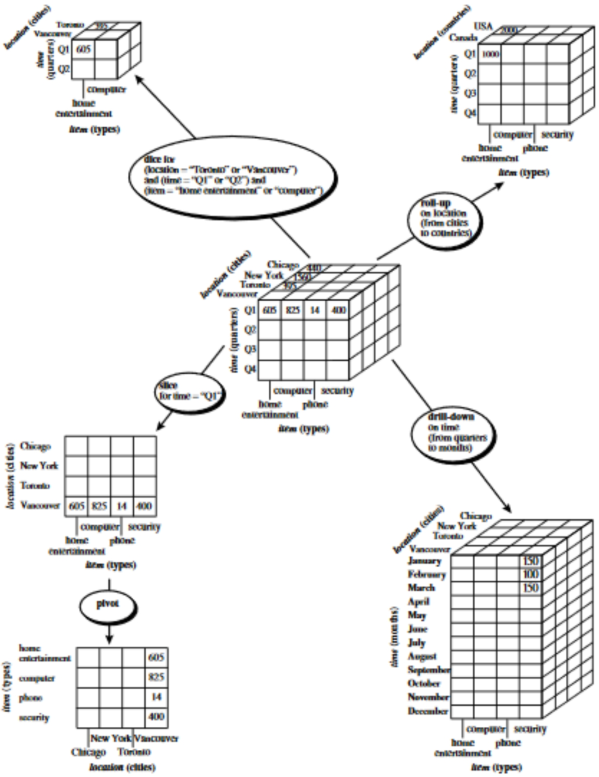

location (cities)
Toronto
Vancouver
395
time (quarters)
Q1
Q2
605
computer
home entertainment
item (types)
dice for
(location = "Toronto" or "Vancouver")
and (time = "Q1" or "Q2") and
(item = "home entertainment" or "computer")
location (countries)
USA
Canada
2000
Q1
Q2
Q3
Q4
1000
time (quarters)
computer
security
home entertainment
phone
item (types)
roll-up
on location
(from cities
to countries)
location (cities)
Chicago
New York
Toronto
Vancouver
440
1560
395
time (quarters)
Q1
Q2
Q3
Q4
605
825
14
400
computer
security
home entertainment
phone
item (types)
slice
for time = "Q1"
location (cities)
Chicago
New York
Toronto
Vancouver
605
825
14
400
computer
security
home entertainment
phone
item (types)
pivot
item (types)
home entertainment
computer
phone
security
605
825
14
400
New York
Vancouver
Chicago
Toronto
location (cities)
drill-down
on time
(from quarters
to months)
location (cities)
Chicago
New York
Toronto
Vancouver
time (months)
January
February
March
April
May
June
July
August
September
October
November
December
150
100
150
computer
security
home entertainment
phone
item (types)

Figure 3.10 Examples of typical OLAP operations on multidimensional data.

Slice and dice: The operation of the slice allows a pick on one dimension of the cube, resulting in a subcube. Figure 3.10 displays a slicing procedure in which the sales data for the dimension time are collected from the central cube using the criterion time = "Q1." The dice procedure determines a subcube by allowing a pick on two dimensions or more. Figure 3.10 displays a central cube dice process centered on the following three-dimensional selection criteria: (location = "Toronto" or "Vancouver") and (time = "Q1" or "Q2"), and (item = "home entertainment" or "computer").

Pivot (rotate): Pivot (also known as rotate) is like a visualization operation point that rotates or revolve the data axes in view to provide an alternative data presentation. Figure 3.10 displays a pivot operation in which the object and the position axes are rotated in a 2-D slice.

Other examples involve rotating the axes in a 3-D cube or transforming a 3-D triangle into a 2-D plane sequence.

Additional OLAP operations

Additional drilling operations provide additional OLAP systems. For instance, drill-across executes queries involving more than one table of facts (i.e., across). The drill-through process uses relative SQL facilities to dig down to its back-end relationship tables from the bottom level of a data cube.

Other OLAP operations may include the ranking of top N or bottom N items in the lists, as well as computing moving averages, growth rates, interests, internal return rates, depreciation, currency conversions, and statistical functions.

OLAP provides mathematical simulation capabilities, including a calculation engine for multi-dimensional derivative ratios, variance, etc., and computational measures. It can generate summaries, aggregations, and hierarchies at every level of granularity and at every intersection of dimensions. OLAP also supports interactive forecasting models, pattern analysis, and statistical analysis. An OLAP engine is a strong data-analysis tool in this context.

OLAP vs. Statistical Databases

Throughout earlier research on statistical databases (SDBs), many of the features of OLAP systems, such as the use of a multi-dimensional data model and definition hierarchies, the identification of measurements with proportions, and the notions of roll-up and drill-down, also occur. A statistical database is a framework developed to support statistical applications. Similarities are rarely discussed between the two forms of structures, mostly due to variations in terminology and application domains.

Yet OLAP and SDB systems have variations to discern. Although SDBs tend to focus on socio-economic applications, OLAP is focused on business applications. Concept hierarchies Privacy problems are a big concern for SDBs. For example, given the socio-economic data summarized, it is controversial to allow users to access the low-level data that responds to the correction. Finally, unlike SDBs, the OLAP systems are designed to efficiently manage large quantities of data.

A Starnet questioning model for multi-dimensional databases

Multi-dimensional database querying can be based on a layout of the starnet. A starnet model is comprise of radial lines emanating from a central point, where each line for a dimension reflects a concept hierarchy. Every level of abstraction in the hierarchy is called a footprint, which reflects the granularities that OLAP operations such as drill-down and example 3.9 Starnet will use for. For the AllElectronics data warehouse, a starnet query model is shown in Figure 3.11. This starnet consists of four radial lines that reflect the Hierarchiesoll-up definition.

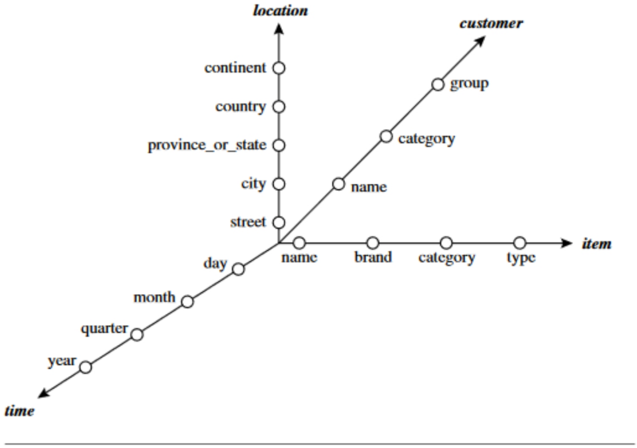


Figure 3.11 Modeling business queries: a starnet model.

For the location, consumer, object, and time dimensions, respectively. That line is made up of footprints which reflect dimensional abstraction levels. The timeline, for example, has four footprints: "day," "month," "fifth," and "year." A concept hierarchy may include a single attribute (like the time hierarchy date) or multiple attributes (e.g., the position hierarchy definition includes the attributes of the street, area, province, or state and country). To analyze the item sales at AllElectronics, users can roll up the time dimension from month to quarter, or, say, drill down from country to city along the location dimension. Concept hierarchies can also be used to generalize data by substituting low-level values (such as "day" for the time dimension) for higher-level abstractions (such as "year"), or by replacing higher-level abstractions with lower-level values to specialize data.

3.4 The Architecture of Data Warehouse

This article provides a structure for market research for building data warehouses. They are often explaining the basic steps involved in the design process.

The architecture of a data warehouse: A system for market analysis

"Using a database, what would business analysts gain? "Second, providing a data center will give a competitive advantage by offering appropriate performance assessment information and making crucial changes to help win over rivals. Second, a data center can improve the efficiency of companies as it can gather information rapidly and easily that accurately represents the organization. Second, a data center enables the management of customer relationships as it offers a clear view of clients and products across all business lines, exits, and markets. Finally, a data warehouse will continuously and efficiently result in cost savings by track-in trends, patterns, and exceptions over long periods. To order to develop an effective data center, we need to understand and evaluate customer needs and create a system for customer analysis. Building a large and complex information system can be seen as constructing a large and complex building, for which the creator, architect, and contractor have different views. These views are combined to create a dynamic structure that reflects the viewpoint of the top-down, business-driven, or owner, as well as the view of the information system from the bottom-up, constructor-driven, or implementer.

It is important to consider four separate views concerning the design of a data warehouse: the top-down view, the view of the data source, the view of the data warehouse, and the view of the business question.

The top-down view helps the data center to pick the appropriate information it needs. This knowledge is in line with current and future business needs.

The view of the data source shows the details that are operating systems record, store, and handle. Such knowledge can be recorded at various levels of depth and precision, from single source tables to comprehensive source tables. Current data modeling approaches, such as the entity-relationship model or CASE (computer-aided software engineering) techniques, are also used to model data sources.

The view of the data center contains tables of information and measurements. It reflects the information contained within the data warehouse, including pre-calculated numbers and counts, as well as information about the source, date, and time of origin added to the historical context.

Eventually, the view of the business question is the view of data from the end user's point of view inside the data warehouse.

Creating and using a data warehouse is a complex activity because it includes expertise in business, technology, and program management. Building a data warehouse includes learning how these systems to store and handle their data, building extractors that transfer data from the operating system to the data warehouse, and building warehouse update software that keeps the data warehouse up-to-date with the data from the operating system. Using a data warehouse requires understanding the nature of the data it holds, as well as knowing and converting business requirements into requests that the data warehouse can fulfill. In regard to technical expertise, data analysts need to learn how to make measurements of quantitative information and to draw evidence-based on assumptions from the data warehouse's information. Such skills include the ability to identify patterns and trends, to extrapolate trends based on experience and search for anomalies or paradigm changes, and to make consistent managerial recommendations based on this research. Finally, program management skills involve interfacing with many technologies, vendors, and end-users to deliver results in a timely and cost-effective manner.

The Data-Warehouse Creation Process

You can build a data warehouse using a top-down approach, a bottom-up approach, or a combination of both. The top-down approach begins with the overall design and scheming. This is useful in situations where the technology is mature and well-known, and where there are simple and well-understood business issues that need to be solved. This method from the bottom upstarts with tests and prototypes. It is useful in the early stage of business modeling and the advancement of technology. It allows and let an organization to move forward at significantly less expense and to evaluate the technology's benefits before major commitments are made. With the combined approach, a company will leverage the expected and strategic complexity of the top-down approach while maintaining the bottom-up approach's rapid implementation and opportunistic application.

From software engineering point of view of, the design and development of a data warehouse may consist of the following steps: preparation, review of specifications, problem analysis, design of the warehouse, system integration and testing, and finally, implementation of the data warehouse. Many large software systems can be developed using two methodologies: the waterfall method or the spiral method. Before proceeding to the next, the waterfall method performs a structured and systematic analysis at each step, which is like a waterfall falling from one step to the next. The spiralling approach requires the rapid development of ever more efficient structures, with short intervals between successive launches. It is considered a good option for the creation of data warehouse, especially for data marts, since the turnaround time is short, improvements can be made quickly, and new designs and software technologies can be adapted or changed in a timely manner.

In general, the design process for the warehouse consists of the following steps:

1. For example, choose a business process to model commands, invoices, shipments, inventory, account management, sales, or the general ledger.
The system is hierarchical and includes several dynamic objects sets, which should be accompanied by a data warehouse model. If, however, the process is departmental and focuses on evaluating one form of business process, then a data mart model should be chosen.

2. Pick the company process food. The grain is the fundamental,

atomic level of data, for example, individual transactions, individual daily snapshots, and so on, to be represented in the fact table for this process.

3. Choose the dimensions that relate to every record of a table of data. Typical measurements are time, object, client, manufacturer, warehouse, form, and status of the transaction.

4. Choose the steps that will fill up every record of the facts table. Typical measurements are numerical quantities of additives, such as dollars sold and units sold.

Since the construction of data warehouses is a complex and long-term undertaking, the extent of their implementation should be clearly specified. The objectives of the initial implementation of the data warehouse should be precise, attainable, and observable. It involves deterring the allocations of time and budget, the sub-set of the enterprise to be modeled, the number of chosen data sources, and the number and types of departures to be represented.

When designing and constructing a data warehouse, the warehouse's initial deployment includes initial installation, roll-out planning, training, and orientation. Often to consider software updates and maintenance. Administration of data warehouses involves data refreshment, data source synchronization, disaster recovery preparation, access control and protection management, data growth management, database output aging, and data warehouse enhancement and extension. Managing the scope includes controlling the number and range of queries, dimensions, and reports, limiting data warehouse size, or limiting schedule, budget, or resources.

There are various kinds of methods for building data warehouses. Data warehouse development tools provide functions for defining and editing metadata repository contents (such as schemes, scripts, or rules), answering queries, recording outputs, and shipping metadata to and from relational system catalogs. Tools for planning and analysis study the effect of changes to the schema and refresh efficiency when adjusting refresh levels or time periods.

3.5 A three-tier Architecture for Warehouse Information

Most data warehouses follow a three-tier architecture, as seen in Figure 3.12.

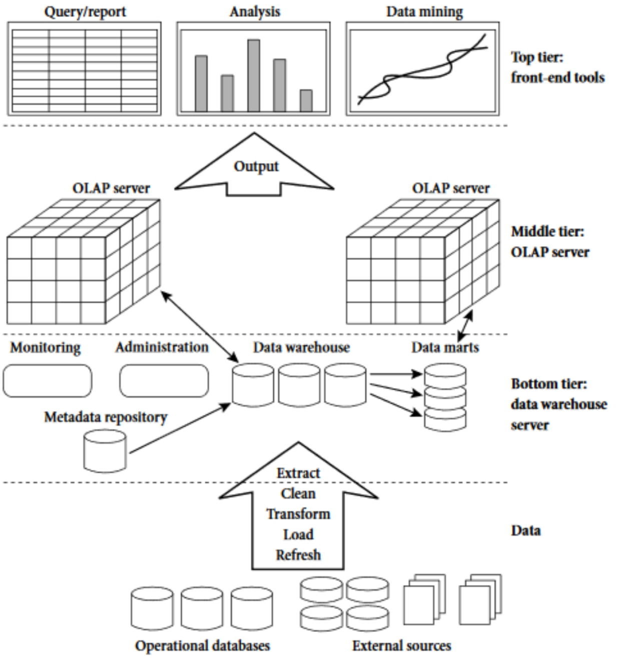

Query/report
Analysis
Data mining
Top tier:
front-end tools
Output
OLAP server
OLAP server
Middle tier:
OLAP server
Monitoring
Administration
Data warehouse
Data marts
Metadata repository
Bottom tier:
data warehouse
server
Extract
Clean
Transform
Load
Refresh
Data
Operational databases
External sources

Figure 3.12 A three-tier data warehousing architecture.

1. The lowest tier is a distribution information server, which is almost always a network of relational databases. Utilities and backend tools and are used to feed data from internal repositories or other external sources (such as customer profile information generated by outside consultants) into the bottom tier. Such tools and applications perform data extraction, cleaning, and integration (e.g., combining similar data from different sources into a single format), as well as loading and refreshing functions to update the data warehouse (Section 3.3.3). The data is collected using the gateways known as computer program interfaces. The underlying DBMS provides a gateway, which allows client programs to produce

SQL code to be executed on a server. Types of gateways include the Windows and JDBC (Java Database Connection) ODBC (Open Database Connection) and OLEDB (Open Link-In and Embedding for Databases). Each tier also includes a metadata archive, which holds data warehouse information and its contents. Section 3.3.4 defines the metadata repository as further.

2. The middle tier is an OLAP server usually implemented using one of these
(1) the OLAP (ROLAP) relational model, that is, the expanded relational DBMS
Maps multidimensional data operations into regular relational operations; or (2) a multidimensional OLAP (MOLAP) model, that is, a special-purpose server that implements multidimensional data and operations director. Section 3.3.5 deals with the OLAP servers.

3. The top tier is the front-end client layer comprising query and reporting tools, analytical tools, and/or data mining tools (for example, trend analysis, prediction, etc.).

From the architecture or structure point of view, there are three types of a data warehouse: enterprise warehouse, data mart, and virtual warehouse.

Enterprise warehouse: An enterprise warehouse gathers all knowledge on issues covering the company as a whole. It offers enterprise-wide data, typically from one or more operating systems or external providers of information, and is cross-functional in nature. It typically contains detailed data and summarized data, and can range in size from a few gigabytes to hundreds of gigabytes, terabytes, or even beyond. For traditional mainframes, supercomputer servers, or parallel architecture platforms, an enterprise data warehouse may be implanted. It requires extensive modeling of the business and can take years to design and build.

Info Mart: A data mart includes a subset of enterprise-wide data that is of interest to a particular user community. The scope is limited to unique chosen subjects. A marketing data mart can, for analysis purposes, confine its subjects to consumer, object, and sales. The data in the data marts appear to be summed up.

Data marts are typically deployed on low-cost, UNIX / LINUX- or

Windows-based departmental servers. A data mart's deployment time is more likely to be calculated in weeks, rather than months or years. Nevertheless, if the architecture and development were not enterprise-wide, it may require complex integration in the long term.

Data Marts may be classified as independent or dependent, depending on the source of the data. Independent data marts are derived from data collected by one or more operating systems or external providers of information, or from data created locally within a particular department or geographic region. Related data marts are directly collected from corporate data warehouses.

Digital warehouse: A digital warehouse is a collection of possible database views. Only any of the possible summary views may be materialized for efficient query processing. A virtual warehouse is simple to install but requires excess space on servers running the database.

"What are the advantages and drawbacks of top-down and bottom-up data-warehouse production approaches? "The top-down architecture of an enterprise warehouse acts as a structural approach and minimizes issues with integration. It is, therefore, expensive, takes a long time to create, and lacks versatility due to the challenge of achieving.

Consistency and consensus for the entire enterprise on a shared data model. The bottom-up approach to planning, creating, and implementing independent data marts offers versatility, low cost, and fast investment returns. But, when combining various fragmented data marts into a single corporate data warehouse, it can lead to problems.

A preferred approach for data warehouse systems implementation is to incorporate the warehouse incrementally and evolvingly, as seen in Figure 3.13. First, a high-level corporate data model is established within a relatively short time frame (such as one or two months) that offers a company-wide, clear, coherent view of data across various subjects and potential usages. While it will continue to be improved in the further development of enterprise data warehouses and departmental data marts, this high-level model will significantly mitigate future integration issues. Second, it is possible to incorporate separate data marts in parallel with the enterprise warehouse, based on the same collection of the corporate data model as above. Second, distributed data marts can be built via hub servers to combine various data marts. Finally, a multitier data center is designed where the corporate center is the sole custodian of all data in the facility, which is then dispersed to the various based data marts.

Back-End Software and Utilities of Data Warehouse

Backend tools and utilities are used by data warehouse systems to fill and update their data (Figure 3.12). The following functions include some resources and utilities:
Data extraction, usually collecting data from various, heterogeneous and external sources Data cleaning, identifying and rectifying data errors where necessary Data transformation, transferring data from legacy or host format to warehouse format Load, sorting, summarizing, consolidating, measuring views, verifying the validity, and building indexes and partitions.

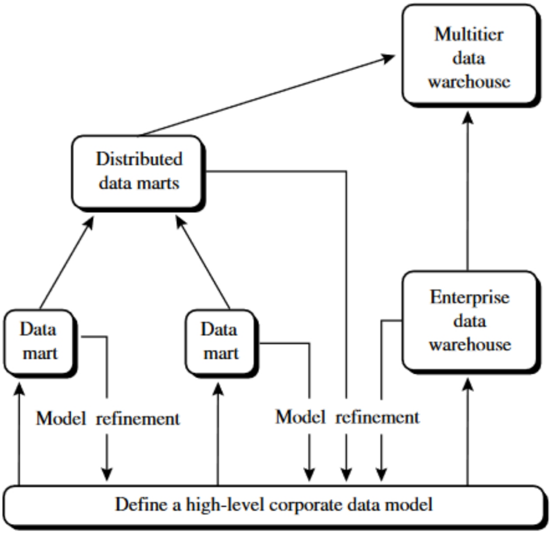

Multitier data warehouse
Distributed data marts
Enterprise data warehouse
Data mart
Data mart
Model refinement
Model refinement
Define a high-level corporate data model

Figure 3.13 A recommended approach for data warehouse development.

In addition to washing, loading, updating, and metadata specification tools, data warehouse systems typically have a strong range of data warehouse management tools.

Software cleaning and data transformation are crucial steps towards improving data quality and data mining performance subsequently. They're listed on Data Preprocessing in Chapter 2. Since we're mainly interested in data warehousing technology aspects relevant to data mining, we won't get into the specifics of the remaining methods and encourage interested readers to review data warehousing technology books.

Repository of Metadata

The metadata is data. When used in a data warehouse, metadata is the data that identifies artifacts in the warehouse. Figure 3.12 revealed a metadata server inside the architecture's lower tier of data warehousing. Metadata is created for the provided warehouse's data names and definitions. Additional metadata will be generated and collected for time-stamping any extracted data, the source of the extracted data, and missing fields introduced through processes of data cleaning or incorporation.

A server with metadata will contain the following:

A summary of the data warehouse structure which includes the warehouse scheme, view, dimensions, hierarchies and derived data definitions as well as the locations and contents of data marts.

Operational metadata, including data heritage (the context of migrated data and the sequence of transformations applied to it), data currency (active, archived or purged), and knowledge tracking (warehouse usage statistics, error reports, and audit trails);

The algorithms used for the summary, which include algorithms for measurement and dimension definition, granularity data, partitions, subject areas, aggregation, summary, and predefined queries and reports. Mapping the data warehouse from the operating environment, which includes source and content databases, gateway details, data partitions, data extraction, cleaning, transition rules and defaults, data update and purging rules, and protection (user authorization and access control)

System configuration data, including indexes and profiles that enhance data access and recovery efficiency, as well as rules for restore, upgrade, and replication cycle timing and scheduling.

Business metadata which includes business terms and definitions, details on data ownership and charging policies.

A data warehouse includes various description levels, of which one form is metadata. Certain types include current (almost always on disk) detailed data, older detailed data (usually on tertiary storage), loosely summarized data, and strongly summarized data (which may or may not be physically stored).

Metadata plays a slightly different role in other data warehouse data and, for several reasons, is significant. Metadata, for example, is used as a directory to help the decision support system analyst find the data warehouse contents, as a guide to locating data as the data is converted from the operating environment to the data warehouse, and as a guide to the algorithms used to summarize the current comprehensive data and the loosely condensed data. Metadata should be persistently maintained and handled (i.e., on a disk).

3.6 From Data Collection to Data Mining

"How does OLAP and Data Warehousing contribute to data mining? "We are researching the use of data storage for information processing, analytical processing, and data mining in this segment. In addition, we are implementing online analytical mining (OLAM), a versatile paradigm that combines OLAP with data mining.

Data warehouse usage

In a wide variety of applications, data centers, and data marts are used. Business managers use the data in data centers and data marts for evaluating data and making strategic decisions. Data warehouses are used in many organizations as an important part of a "closed-loop" plan-execute-assessment feedback mechanism for corporate management. Information warehouses are commonly used in banking and financial services, consumer goods and retail delivery, and industrial controls, such as demand-based production.

Usually, the longer a data warehouse was in operation, the more it grew. Such evolution happens through a variety of stages. The data warehouse is initially primarily used for report generation and answering predefined queries. This is used to evaluate condensed and detailed data gradually, where the findings are presented in the form of reports and charts. The data warehouse is subsequently used for strategic purposes, per-forming multidimensional analysis, and advanced slice and dice operations. Ultimately, using data mining techniques, the data warehouse can be used for information creation and strategic decision making. The data warehousing tools can be classified in this context into access and retrieval tools, database monitoring tools, data processing tools, and data mining tools.

Data Business users need to have the means to know what resides in the data warehouse (through metadata), how to access data warehouse information, how to use analytical tools to analyze material, and how to impersonate the results of such research.
There are three types of applications in the data warehouse: information processing, analytical processing, and data mining:

Information analysis facilitates querying, essentially reviewing data and reporting using crosstabs, tables, maps, or graphs. A new trend in the processing of data warehouse information is to develop low-cost Internet-based access tools that are then incorporated with Web browsers.

Analytical processing facilitates simple operations in OLAP, including slice-and-dice, drill-down, roll-up, and pivot. Usually, it works in both condensed and comprehensive ways of historical data. Multidimensional data analysis of data warehouse data is the key strength of online analytical processing over information processing.

Data mining facilitates the exploration of information by discovering secret patterns and associative, constructing analytical models, conducting classification and prediction, and presenting the effects of mining using visualization instruments.

"How does data mining apply to the collection of information and web analytics? "Data retrieval can find valuable information based on queries. How- ever, the answers to these queries represent knowledge that is stored directly in databases, or that can be placed by aggregate functions. They do not represent in the database sophisticated trends or regularities buried. Therefore, the retrieval of information is not data mining.

Online analytical analysis comes to a step closer to data mining as it can extract information from user-specified subsets of a data warehouse, presented at various granularities. Such descriptions are equivalent to the descriptions of the class/concept discussed in Chapter 1. This poses some important questions because data mining systems can also mine generic class/concept descriptions: "Do OLAP systems perform data mining? Are OLAP systems actually the systems for data mining? OLAP's and data mining functionalities can be described as disjoint: OLAP is a data summary/aggregation tool that helps simplify data processing, while data mining allows for the automated discovery of latent patterns and interesting insights concealed in vast volumes of information.

The purpose of OLAP tools is to simplify and promote interactive data analysis, while the aim of data mining tools is to get the automate system as much of the process or method as possible while still allowing users to direct it. In that sense, data mining goes beyond conventional online analytical analysis one step further. An alternative and broader definition of data mining, in which data mining includes both data classification and data modeling, may be adopted.

Since OLAP systems are able to present general data descriptions from data warehouses, OLAP functions are essential for user-driven data summary and comparison (drilling, pivoting, slicing, dicing, and other operations). Such are, although minimal, functionalities in data mining. However, according to this view, data mining covers a far wider range than simple OLAP operations as it conducts not only data description and comparison but also correlation, classification, prediction, clustering, time series analysis, and other data analysis tasks. The study of data contained in data warehouses is not limited to data mining.

It may analyze existing data at more detailed granularities than the data summarized in a data warehouse. This can also analyze transactional, temporal, textual, and interactive data, which are difficult to model using current multi-dimensional database technology. While in this context, data mining covers a broader spectrum than OLAP regarding the functionality of data mining and the complexity of the data handled. Data mining required to have broader applications because it involves more automated and deeper analysis than OLAP.

Data mining can help business managers identify and attract better-matched consumers, as well as gain valuable business insights that can help boost market share and increase profits. Additionally, data mining can help managers understand the characteristics of the customer community and develop appropriate pricing strategies accordingly, accurate item bundling based not on assumptions but on real item groups extracted from customer purchase patterns, minimize promotional expenses and at the same time improve the overall net effectiveness of promotions.

From Analytical Web Processing to An Online Analytical Mining

Substantial effort has been carried out in the field of data mining on various platforms for data mining, including transaction databases, relational databases, spatial databases, text databases, time-series databases, flat files, data warehouses, and so on.

Online analytical mining (OLAM) (also known as OLAP mining) integrates online analytical processing (OLAP) in multi-monthly databases with data mining and mining knowledge.

OLAM is especially important among the many different paradigms and architectures of data mining systems, for the following reasons: High data quality in data warehouses: Many data mining tools need to operate on distributed, reliable and clean data requiring costly data cleaning, data replication, and data transformation as pre-processing steps.

A data warehouse built through such pre-processing serves as a valuable source of high-quality data both for OLAP and for data mining. Note that data extraction can also serve as a valuable tool for data cleaning and data integration. Available data processing infrastructure for data warehouses: robust data processing and data analysis infrastructure for data warehouses, including access, integration, consolidation, and transformation of multiple heterogeneous databases, ODBC / OLE DB links, online access, and support facilities, and Rather than constructing anything from scratch, it is wise to make the best use of the existing infrastructures. OLAP-based exploratory data analysis: Exploratory data analysis requires efficient data mining.

Often a user will want to traverse a database, select portions of relevant data, analyze them at different granularities, and present knowledge edge/results in various forms. Online analytical mining offers data mining facilities by drilling, pivoting, filtering, dicing, and slicing on a data cube and on certain intermediate data mining results on various subsets of data and at various levels of abstraction. This will greatly enhance the power and flexibility of exploratory data mining, in conjunction with data/knowledge visualization instruments. Online collection of data mining functions: Sometimes, a user may not realize what kind of information they want to exploit.

By integrating OLAP with multiple data mining features, online analytical mining gives users the flexibility to dynamically select desired data mining functions and swap data mining tasks.

Electronic Analytical Mining System

An OLAM server does analytical mining in data cubes in the same way as an OLAP server does online analytical processing.

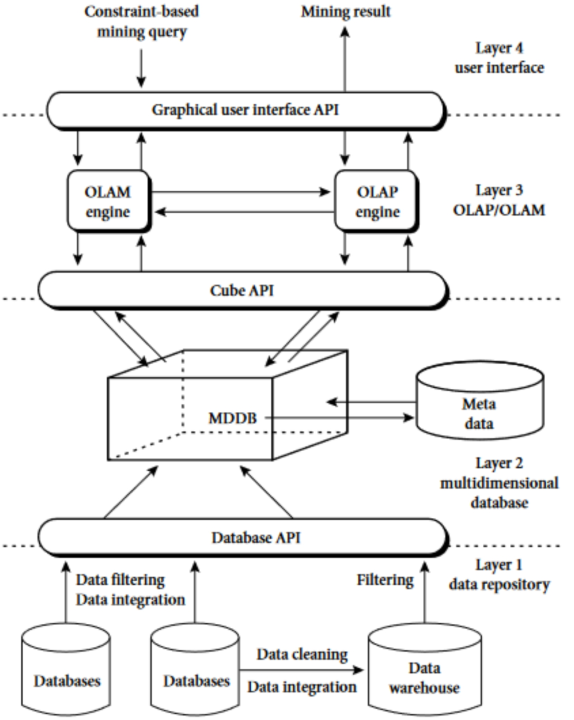

Constraint-based mining query
Mining result
Layer 4 user interface
Graphical user interface API
OLAM engine
OLAP engine
Layer 3 OLAP/OLAM
Cube API
MDDB
Meta data
Layer 2 multidimensional database
Database API
Data filtering
Data integration
Filtering
Layer 1 data repository
Databases
Databases
Data cleaning
Data integration
Data warehouse

Figure 3.18 An integrated OLAM and OLAP architecture.

The integrated OLAM and OLAP architecture are seen in Figure 3.18, where both OLAM and OLAP servers accept online user requests (or commands) through a graphical user interface API and work with the data cube via a cube API in data analysis. It uses a metadata directory to guide data cube entry. The data cube can be developed by accessing and/or integrating multiple databases via an MDDB API and/or filtering a data warehouse via an OLE DB or ODBC connection-supporting database API. As an OLAM server can perform multiple data mining tasks, such as concept description, correlation, prediction, classification, clustering, time series analysis, and so on, it typically consists of various integrated data mining modules and it is more sophisticated than an OLAP server.

3.7 Data Mining Distributed Systems

Modern distributed computing may derive from the efforts made in the 1990s to create "Beowulf" computer clusters and from early ad hoc computing. Beowulf computer clusters were standard servers or even networked desktop computers. On the machines, the program then interacted closely across the network to break the work to be performed around the devices. Early ad hoc computing efforts, including distributed.net and SETI@Home, used idle home computers to search the radio astronomy data for cryptography keys and alien radio signals. None of these projects had the in-house computing resources to solve the problem, nor had enough budget to do so. Nevertheless, the problems were simple enough that a large number of computers with little communication between them could make substantial progress. Home computers running the distributed.net or SETI@Home program have contacted the main server to do a chunk of work (keys to search or radio signal data to examine).

Classical supercomputers were huge, very expensive machines containing powerful hardware and processors specialized in them. These supercomputers had features such as multiple central processing units and vector hardware included in their specialized hardware. Vector hardware is used on multiple pieces of data to perform identical operations. For example, components X and Y are in a two-dimensional vector. Components add a number of vectors, so delivering a single instruction that adds both X and Y at the same time will lead to doubling the processing speed.

System prices have shrunk significantly. Both high-core / large-memory (massively parallel processing [MPP] systems) and moderate-system clusters allow the realization of much larger data mining projects. The only solution for Big Data analytics is to move the analytics to the data; it is impractical to move the data to the analytics due to the time required for the data transfer.

Harder issues — problems that consume much greater volumes of data, with much higher variable numbers — can now be considered for review. It is possible to split cluster computing into two major classes of distributed computer systems. The first is the database, which has been a staple for decades in data centers. The second is the distributed file system, now dominated by Hadoop.

DATABASE SYSTEM

The Relational Database Management System (RDBMS) or simply database has been around since the 1970s and was the most common place to store organizations-generated data until recently. Database systems are provided by a number of huge vendors as well as open-source projects. Traditional RDBMSs are designed to serve databases far larger than the usable memory or storage on a single device. There are now a number of different databases from those early days which serve special purposes for organizations around high-performance data mining and big data analytics.

The development of in- memory databases (IMDBs) began in the 1990s. IMDBs are now a common tool used to speed up mission-critical data transactions for banking, e-commerce, social media, IT, and other industries. The principle of be-hind IMDB technology is straightforward — holding data in memory and performance improvements, not on the disk. However, IMDBs have notable disadvantages, namely the increased memory expense and uncertainty.

In the 1980s, MPP (massively parallel processing) systems started emerging from conventional DBMS technologies. MPP databases are intended to serve many of the same operational, transactional, and analytical purposes as previous commercial database generation but offer performance, availability, and scalability features designed to handle large data volumes while using standard user interfaces. The MPP databases are positioned as the most direct update for data warehouses (EDWs) for organizations. MPP database technology usually involves commodity clusters or specialized servers, holding data on multiple hard drives.

A big advantage of computing databases is the time saved in moving the data to the analytics. The time required to transfer the data and the hardware resources to process it efficiently once it is transported makes the strategy inefficient in a genuinely big data problem. Using software that can transfer analytics to the data and process it in place, using the vast computational resources a distributed database provides can result in faster models and shorter running times compared to non-distributed computing systems.

FILE COMPUTING SYSTEM

There are many options in choosing a file system computing platform, but the market is rapidly consolidating on Hadoop with its many file system compatible distributions and tools, such as Hive, MapReduce, and HBase.

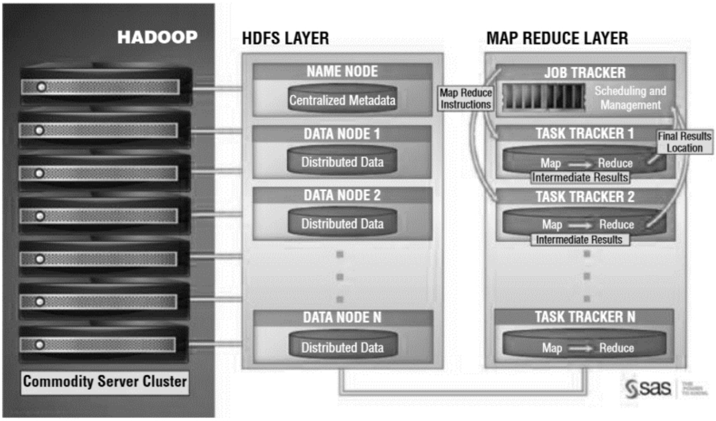

HADOOP
HDFS LAYER
MAP REDUCE LAYER
NAME NODE
Centralized Metadata
Map Reduce Instructions
JOB TRACKER
Scheduling and Management
DATA NODE 1
Distributed Data
TASK TRACKER 1
Final Results Location
Map → Reduce
Intermediate Results
DATA NODE 2
Distributed Data
TASK TRACKER 2
Map → Reduce
Intermediate Results
DATA NODE N
Distributed Data
TASK TRACKER N
Map → Reduce
Commodity Server Cluster
sas

Figure 2.1 Graphical illustration of a Hadoop System

By Doug Cutting and Mike Cafarella, Hadoop was formed. (See Figure 2.1.) The word "Hadoop" does not really reflect an acronym or a reference to something. It came from the name given to a stuffed yellow elephant by Cutting's friend. It was initially developed in 2004 based on the work done on Nutch2 by Cutting and Cafarella and a paper published by Google, which introduced the MapReduce paradigm for processing data on large clusters. Hadoop became an Apache project at the top level in 2008 and was used by several large data companies like Yahoo!, and the New York Times.

A current trend is for all available data to be stored in Hadoop. Hadoop is attractive because it can store and handle very large quantities of commodity hardware data, and can easily expand with incremental cost by adding hardware resources. Systems have historically been designed based on projected data volumes, without the assumption that data will accumulate in perpetuity.

Hadoop has made data mining on a wide scale feasible and is potentially a major distinguisher for competitive advantage. Those who can effectively leverage the importance of historical data will gain a significant future advantage.

Hadoop is the front runner for storing large amounts of historical data. But this data is rarely (probably never actually) in a suitable form for data mining.

Also, in Hadoop, other data repositories are used for many problems to increase the data. For example, credit card transactions may be stored in Hadoop, but information about the cardholder account may be stored and maintained in a traditional database. The data survey phase involves the identification of the data and the data sources to be used in the modeling process. For data mining, the data must then be compiled, condensed, and processed. This research would usually be performed in Hadoop since the cost of computing per device is lower than MPP or RDBMS. This hybrid approach to data storage will probably be common practice for many years until either Hadoop matures to database level or another technology emerges that makes both databases and Hadoop less desirable.

Cloud computing can be used in any capacity as it can easily supply new servers or break down servers that are no longer required. But not all of the high d modifications are usable, such as specialized networking hardware. Their flexibility makes it possible to switch quickly between simple ad hoc computing and coordinated computing, and even to mix the models on the fly.

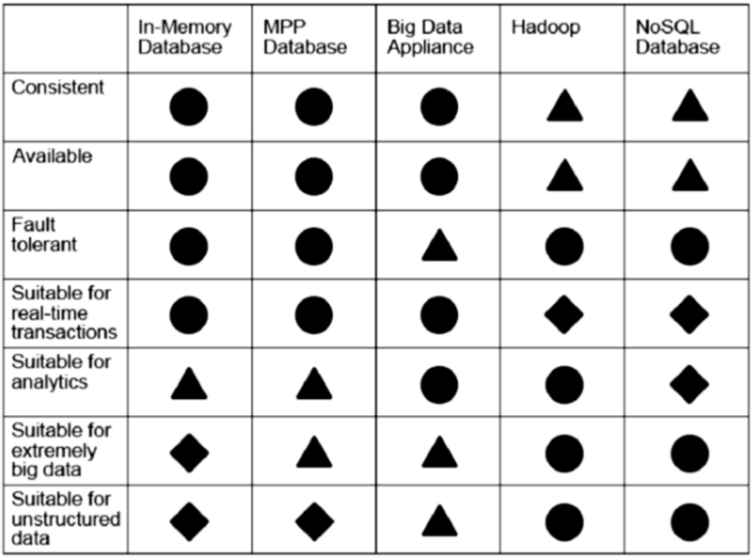
CONSIDERATIONS

Here are some issues to ask as you find your data mining platform to be high performing.

- What is your data size now?
- What is your data's projected growth rate in the coming years?
- Are the data you mostly store structured or unstructured data?

- Which data movement is needed to complete your data mining projects?
- Which percentage of your data mining projects can be solved on one machine?
- Is your data mining program a good fit for your computing environment?
- What are the key issues from your customers about the new system?

Compares a number of big data technologies in Figure 2.2. The figure illustrates the various types of systems and their comparative strengths and weaknesses. The IT department will undoubtedly make the buying decision for the data mining computing platform, so it must first consider the challenges that are already being addressed and the issues that need to be fixed in the immediate future. Understanding network trading FFS and the organization's needs, all communicated in a straightforward way, should contribute to the best outcome for the enterprise as a whole and the individual stakeholders.



	In-Memory Database	MPP Database	Big Data Appliance	Hadoop	NoSQL Database
Consistent	●	●	●	▲	▲
Available	●	●	●	▲	▲
Fault tolerant	●	●	▲	●	●
Suitable for real-time transactions	●	●	●	◆	◆
Suitable for analytics	▲	▲	●	●	◆
Suitable for extremely big data	◆	▲	▲	●	●
Suitable for unstructured data	◆	◆	▲	●	●

Figure 2.2 Comparison of Big Data Technologies

The symbols in Figure 2.2 have the same interpretation as:

- Circle: meets expectations that are widely held.
- Triangle: Potentially fulfills standards commonly held.
- Diamond: Fails to meet standards commonly held

Big data/data mining projects have to address how data is moving through the whole end-to-end process.

The computing environment is key to success: There are four essential tools to include in your computing environment: network, disk, central processing units (CPUs), and memory. Time to solve goals, expected volumes of data, and budget will direct your computational resource decisions. Appropriate computing platforms for data analysis depend on many dimensions, notably data volume (initial, work set, and output data volumes), data access patterns, and analysis algorithms. These will vary depending on the data analysis phase.

3.8 Analytics Tools Used for Data Mining

People need resources to solve challenges and find ways of generating value in the analytics forum, which often begins very early in deciding which tool to use. This is the third factor prior to the start of research. Some I've got a lot of familiarity with and actually contributed to their growth. Others I used as tools, just as you or your teams will, and some I didn't use personally, but they should also be considered. I have tried to ensure a balanced evaluation of the strengths of each instrument and its weakness.

WEKA WICH

Weka (Waikato Environment for Information Analysis) is an offering of open source data mining, completely implemented in Java, and developed primarily at the University of Waikato, New Zealand. Weka is notable for its wide variety of highly sophisticated training algorithms, its graphical user interface (GUI) workflow, and its implementation of tools for data visualization. Weka allows users to access its sophisticated data mining routines through a GUI designed for productive data analysis, a command-line interface and a programming interface (API) for Java applications. Weka does not well scale for big data analytics, however, as it is limited to the available RAM resources, typically on one machine. Users with 64-bit operating systems will have access to much larger RAM resources, but Weka's documentation directs users to pre-processing and filtering algorithms for sampling big data before analyzing them. Since many of Weka's most powerful algorithms are unavailable in other environments without the development of custom software, sampling may be the best practice for use cases that Weka requires. Weka has made strides towards multithreading and simple multitasking in recent releases. In Weka 3.6, several classifiers that simultaneously train models on multiple cross-validation folds, and the Weka server implementation enables data mining tasks to be run simultaneously on a single machine or cluster. Weka routines also provide advanced analytics components for two Java-based large data analysis environments: Pentaho and MOA.

Weka is a good choice for people who may not have expertise in programming Java and need to get started quickly to prove value. During a graduate school survey course, I used Weka about data mining tools. It was a few years after Weka's use of Java had been completely rewritten. The design provided me with major productivity benefits because I wasn't a very professional Java programmer. I was able to analyze data in the programming language with far less investment in startups.

JAVA LANGUAGES AND JVM

Java and several other languages that run on the Java Virtual Machine (JVM) are common choices for organizations looking to design custom analytics platforms from scratch. Java offers considerable development advantages over lower-level languages for massive, concurrent, and networked applications, with no unnecessary performance sacrifices, particularly in the realm of big data analytics. Although languages running directly on native hardware, particularly FORTRAN and C, continue to outperform Java in RAM and CPU-bound calculations, the Java platform's technological advancements have brought its performance almost in line with native languages for input/output and network-bound processes, such as those at the heart of many open source big data applications. In the 2008 and 2009 TeraByte Sort Benchmark, the perhaps most recognizable Java-based big data system, Apache Hadoop, beat out many other technologies. It was the first Java-based technology to win the respected competition for benchmarking.

The benefits and disadvantages of Java as a general-purpose programming language have been widely discussed in many forums, and will only be addressed briefly here. Analytic application developers have always avoided Java in the past because the exhaustive run-time library was inadequate for numerical routines, and the JVM platform's memory management caused unreasonable slowdowns, to mention only two factors. As the scale and complexity of analytical applications grew dramatically, development time became a more serious concern, and memory and CPU-bound performance became less so. Java's strengths arise from its development efficiency in the specific context of building data mining applications: its rich libraries, many application frameworks, inherent support for competitiveness and network communications, and a pre-existing open-source code base for data mining features, such as the Mahout and Weka libraries. While development advantages also stem from the JVM platform at some cost to performance. The JVM provides memory management, memory p ling, and automated handling of exceptions, in addition to portability.

Scala and Clojure are new languages that also run on the JVM, and are used for applications in data mining. Scala is an open-source language developed by the École Polytechnique Fédérale de Lausanne in Switzerland. It was released in year 2003 for the first time and aimed to improve on Java.

Scala is used to building data mining applications because it encompasses Java's development efficiencies — through interoperability with the Java Runtime Environment — and adds a more concise syntax to functional programming. Scala allows developers to switch between functional programming and object-oriented (OO) programming paradigms while grammar and syntax in Java tend to enforce OO design. In data mining applications, functional languages are considered good, as they treat memory differently from OO and procedural languages. For example, a major concern in multithreaded procedural and OO applications is to prevent multiple threads from altering the same variable at the same time. Functional programming languages prevent this by allowing variables to never shift. While being a less popular programming style, functional programming is a powerful method not to be mocked for. Scala is becoming increasingly popular and has been used to introduce major open sources projects, such as Akka and Spark. Akka is a toolkit to build large, parallel, and distributed JVM applications, and Spark is a high-performance data mining environment from the Berkeley AMP Lab University of California. Also, Scala is used commercially by FourSquare, the Guardian, LinkedIn, Novell, Siemens, Sony, Twitter, and others for various purposes.

Rich Hickey wrote Clojure, and it was published in 2007. It is a practical programming language, and a Lisp programming language dialect, with additional features for scripting and competition. Though not specifically designed for big data applications, these intrinsic features make Clojure an excellent candidate for building software for data analysis. Clojure defaults on immutable data structures, removing a whole category of concerns about thread safety design compared to procedural and OO languages. Clojure handles data in a rather holistic way, providing developers with several convenient ways of processing their extensive collection of data structures and inheriting Lisp's code definition as data. This approach to data sequences eliminates another class of common programming errors associated with boundary conditions, especially as compared to C and FORTRAN. Even Clojure inherits from Lisp's simple syntax, lazy evaluation, and highly efficient macro facility. The macro facility allows the creation of domain-specific languages (DSLs), and DSLs have already been established for integration with SQL (ClojureQL) and Hadoop (Cascalog). When Clojure combines the unique strengths of the JVM platform with those of access to Java libraries, it becomes a formidable asset for general-purpose programming. Clojure's first adopters generally were smaller, technologically, or analytically advanced firms and startups. But larger corporations and institutions like Citigroup and the Max Planck Institute have also reported using Clojure.

R

R is an open-source programming language of the fourth generation, designed for statistical analysis. R evolved out of the commercial S language developed at Bell Laboratories from the late 1970s onward. R had been ported by researchers at the University of Auckland, New Zealand from S and SPLUS — another commercial implementation now licensed by TIBCO — and first appeared in 1996.

R is very common in academia, where I was first exposed. I used it to play with new analyzes that I had little time or experience to code myself and complete assignments to homework. Even at the time, the program had superior graphing capabilities to other ones that I had used. In graduate school, and while working in the U.S., I used this graphing facility. Census office. Given the fact that these graphing facilities remain in R, it has lost its competitive edge to other applications.

R has grown to a prominent position in the broad and growing community of data science, enjoying widespread use in academic settings and increasing private sector acceptance. Large corporations are known to use R for a number of purposes, such as Bank of America, Google, Intercontinental Hotels, Merck, Pfizer, and Shell. The September 2013 TIOBE general survey of programming languages placed R 18th in overall popularity in the development language, far outstripping S and S- SPLUS, and close to commercial solutions such as SAS (at 21st) and MATLAB (at 19th) (TIOBE, 2013). 1 R is extraordinarily customizable, too. There are thousands of R extensions, up from about 1,600 in 2009. Extension services cover everything from speech processing, genomics, text mining.

R also includes outstanding graphics, free and polished integrated software (IDE) environments.

R is undergoing rapid growth in popularity and functionality, and many of the memory issues that made it difficult for previous versions of R to deal with larger data have been resolved. Significant development work is currently underway to overcome remaining memory constraints through cluster computing and Hadoop interfaces. R's short update cycle made the work available to R consumers through official packages such as snow, multicore, and parallel, and younger projects such as R Hadoop and RHIPE. Such packages, aside from R hadoop and RHIPE, are not intended for truly big data analysis, but for CPU- and memory-intensive calculations of the "embarrassingly parallel" type, mostly performed via the lapply) (method, where a function is applied to all the elements of a list.

Snow is used for executing R calculations on a computer cluster. This uses sockets, MPI, PVM, or NetWorkSpaces to communicate between cluster nodes. The snow kit uses conventional master-worker parallel architecture, keeps memory results on the master node, and includes arguments to fit in memory snow function calls. The kit, thus, being memory-bound, is mainly used for CPU-intensive computations such as simulations, bootstrapping, and some algorithms for machine learning.

Multicore is a simple-to-install and-implement package which splits sequential R processes running on a single POSIX-compliant machine (OS X, Linux, or UNIX) into multithreaded processes running on the same machine. Processes performed using multicore concurrently are limited to single-machine RAM and other shared memory constraints.

A parallel is a parallel execution package which comes in R 2.14 and higher as usual. This extends parallel execution out-of-the-box to cluster level for POSIX-compliant operating systems, and multiprocessor level for Windows. Parallel shares a great deal of snow and multicore syntax, functionality, and limitations, and can use objects from the snow clusters.

R Hadoop and Rhipe allow programmatic Hadoop access, using the R language. On a Hadoop cluster, Map and Reduce tasks can be run from R.

PYTHON

Python was designed to be an extensible, high-level language with a large standard library and a simple, expressive syntax from its inception in the late 1980s. Python can be used interactively or programmatically and is commonly used for scripting, numerical analysis, and general-purpose OO and development of Web applications. Python is a language interpreted and usually performs slower computationally intensive tasks than natural, compiled languages and JVM languages. However, through several documented APIs, Python programs can call compiled, native FORTRAN, C, and C++ binaries at run-time. Python programs can be multithreaded, and like JVM languages, memory management, and exception handling is carried out by the Python platform, which relieves the developer from that responsibility. The Python language was also ported to the JVM framework in a Python project.

Combined with its many databases, math, and graphics libraries, Python's general programming strengths make it a good fit for projects in data exploration and data mining problem domains. Python's inherent file handling and text manipulation capabilities, combined with its ability to connect to most SQL and NoSQL databases, allow Python programs to fairly easily load and process raw and tabular data. The Scipy and Numpy libraries can be used to implement mathematical and statistical models, which provide good support for linear algebra, numerical analysis, and statistical operations. New Python libraries such as Orange and Pattern provide high-level data mining APIs, and the Matplotlib library can generate complex and nuanced visualizations of data. Python can also use the Hadoop Streaming utility to write maps and reduce directives for MapReduce jobs.

Recently, with the development of the scik learning toolkit, Python has become much more used in data mining and data science communities. Currently, the toolkit is releasing 0.14 but provides a range of useful algorithms and manipulation techniques. Python's scikit has no user interface beyond the programming IDE, which allows flexibility for al-users but makes the initial learning curve steep and reduces productivity for routine tasks.

SAS

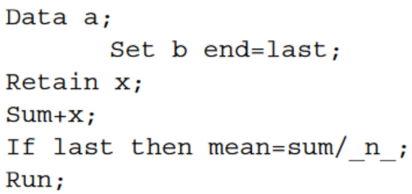
SAS is the market leader in analytics software. SAS not only had a sig can’t market share in the 2013 IDC survey ("Worldwide Business Analytics Software") but also had a larger market share than the next 16 vendors combined. The SAS System was first marketed in 1976, and two of the four cofounders of the company are still alive. And John Sall and Jim Goodnight. The SAS Framework is divided into a variety of product areas, including information, research operations, data processing, data-access engines, and business intelligence (BI). SAS / STAT ®, SAS ® Enterprise MinerTM, and the SAS Text Analytics Suite are the most important tools for this book. The most recent Forrester Wave and Gartner Magic Quadrant have ranked SAS as the leading vendor in predictive modeling and data mining. These ratings show the breadth and depth of the software's capabilities. There is no analytical challenge in my experience that I wasn't able to accomplish with SAS. Worked for the company for around a decade, I have a much better understanding of the development team's hard work and commitment, and the incredible lengths they go to ensure the quality of the product, satisfy customer demands, and foresee potential customer needs.

The SAS system is divided into two main areas: analytical procedures and the language of the fourth generation that allows data manipulation by users. This is known as Phase DATA.

One of SAS's unique accomplishments is backward compatibility, which is maintained after release. I saw it at the U.S. first hand Census office. We had many SAS programs well over 15 years old, and with each upgrade to SAS, they delivered the same verifiable results. This was essential because the original author of the computer programs had frequently moved on to other departments or, in some cases, did not work at the Census Bureau anymore.

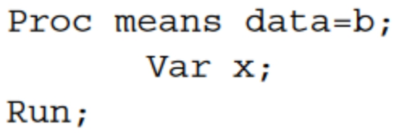
SAS uses a robust scheme to achieve the highest value in programming while being responsive to individual customer environments. The procedure, or PROC as it is commonly referred to, is a module designed for specific analysis, such as the REG procedure for building regression models or the UNIVARIATE procedure for carrying out descriptive statistics on a single variable; Let's take a task of computing a mean for a given variable, in a simple example.

In any programming language, this can be done trivially because the mean is simple: Σ1n I / n . For code like this, this can be computed in the data step:



```
Data a;
      Set b end=last;
Retain x;
Sum+x;
If last then mean=sum/_n_;
Run;
```

This example shows where a new dataset from data set b will be created. The retain statement tells SAS to keep the values when moving to a new observation (row); the if statement tells SAS to divide the sum by the number of observations stored in the automatic variable n if this is the last observation in the data set; This code isn't hard to write and explain, but for SAS users there is an alternative approach that's better:



```
Proc means data=b;
      Var x;
Run;
```

In addition, this code is also easier to read, using code written for a specific purpose rather than a general-purpose code, and I didn't have to make a copy of the data just to add a new column. The procedure also handles and documents a number of behaviors that become very important when developing your own software (e.g., working with ties, missing values, or other issues regarding data quality). Only the SAS / STAT product has 82 procedures. Hundreds of procedures do specific analyzes, literally. Documentation is one major advantage of SAS over other software packages. SAS / STAT has over 9,300 pages of reports and over 2,000 pages of Business Miner's.

Whenever possible the SAS system processes data in memory and then uses systems page data efficiently at any scale. SAS has made significant improvements in its architecture in recent years to take full advantage of the processing power and declining price per FLOP (fl).

Chapter 4: Data Mining And Machine Learning

We are overwhelmed by the details. The volume of data in the world, in our lives, continues to continue to grow — and there's no end in vision. Personal computers that we make it too easy to save stuff that we'd trashed previously. Inexpensive and modest multi-gigabyte disks make it too easy to delay decisions about what to do with all of this — we're just buying another disk and keeping it. Ubiquitous gadgets keep track of our decisions, our shopping choices, our financial practices, our comings, and goings. We swipe our way around the universe, and every record is swiped into a database. We're overwhelmed by knowledge on the World Wide Web; meanwhile, every choice we make is registered. And all these are purely personal choices: they have countless parallels in the commercial and industrial world. We'd all bear witness to the growing distance between data generation and our understanding of it. When the number of data increases, the proportion of this that people understand declines inexorably, alarmingly. Hidden concealed in all of this data is information, potentially valuable information, which is rarely made clear or exploited.

This book revolves around searching for data trends. There is nothing new in this respect. Since human life began, people have been searching for trends in data. Hunters are looking for trends in animal migration behavior, farmers are looking for trends in crop production, politicians are looking for patterns in voting sentiment, and lovers are looking for patterns in the responses of their partners. A scientist's job (like a baby) is to make data sense, to discover the patterns that control how the physical world functions, and to encapsulate them into hypotheses that can be used to anticipate what will happen in new circumstances. The role of the entrepreneur is to find opportunities, that is, behavior patterns that can be turned into an exploit, a profitable business.

The data is processed electronically in data mining, and the search is automated-or at least enhanced-by computer. But this isn't entirely new. Economists, statisticians, forecasters, and communications engineers have long been working with the concept that data patterns can be automatically searched for, detected, checked, and used for prediction. What's fresh is the dramatic rise in data pattern-finding opportunities. Throughout recent years, the unbridled growth of databases, databases on such daily things as consumer preferences, placed data mining at the forefront of emerging business technologies. The amount of data accumulated in the world's databases has been estimated to double every 20 months, and while it would certainly be difficult to quantify this number in any quantitative context, we can all relate qualitatively to the speed of development. The opportunities for data mining are growing as the influx of data swells and computers that can perform the search become commonplace. When the planet increases in complexity, overwhelming us with the data that it generates, data mining is our only chance to elucidate the trends that underlie it. Smartly evaluated data provides a valuable tool. This can lead to new ideas and strategic advantages in commercial settings.

Software mining involves solving problems through the study of software already existing in databases. Suppose the issue is fickle consumer loyalty in a highly competitive environment if one takes a well-worn example. A customer preference database, along with client profiles, holds the key to this issue. Former consumer behavior patterns can be analyzed to recognize features that differentiate those likely to move products and those likely to stay loyal. When these traits are detected, they can be put to work to classify customers present who are likely to jump ship. This category may be selected for special care, which is too costly to be extended to the entire consumer base. Most importantly, it is feasible to use the same strategies to recognize consumers who may be drawn to another service that the company provides, one that they do not actually like, and qualify them for promotional deals that support that service. Data is the raw material that drives business growth in today's highly competitive, customer-centered, service-oriented economy — if only it can be mined.

The method of discovering patterns in the data is known as data mining. The method has to be either automatic or (more commonly) semiautomatic. The trends that have been discovered must be significant in that they offer some advantage, usually an economic gain. The data are typically present in significant quantities.

Why do trends convey themselves? Useful trends allow us to make non-trivial predictions about new data. For the representation of a pattern, there are two extremes: as a black box whose innards are essentially nonsensical, and as a transparent box whose design exposes the pattern structure. We believe both are making good predictions. The actual difference is whether or not the patterns being mined are interpreted in terms of a framework that can be checked, argued on, and used to guide future decisions. We consider these patterns structural since they specifically capture the decision structure. To put it another way, they help clarify something about the results. Now, we can actually tell you what this book is about. It is about techniques used to recognize and explain structural trends in data. Many of the methods we discuss have developed within an area called machine learning. Yet let's look first at what the structural trends are.

Describe Structural Trends

What do structural trends mean? Why are you explaining those? And what form does the input take? Instead of attempting formal, and essentially sterile, definitions, we will address these questions by example. Earlier in this chapter, there will be plenty of cases, but let's discuss one thing right now to get a perception of what we're talking about.

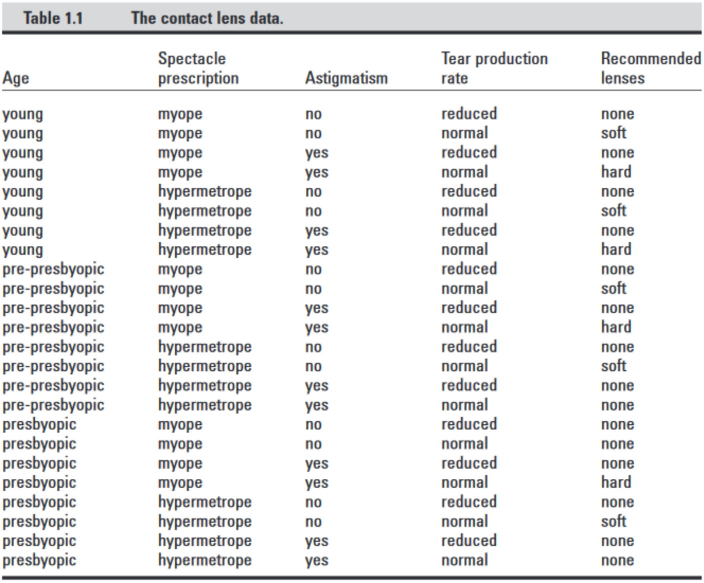


Table 1.1 The contact lens data.

Age	Spectacle prescription	Astigmatism	Tear production rate	Recommended lenses
young	myope	no	reduced	none
young	myope	no	normal	soft
young	myope	yes	reduced	none
young	myope	yes	normal	hard
young	hypermetrope	no	reduced	none
young	hypermetrope	no	normal	soft
young	hypermetrope	yes	reduced	none
young	hypermetrope	yes	normal	hard
pre-presbyopic	myope	no	reduced	none
pre-presbyopic	myope	no	normal	soft
pre-presbyopic	myope	yes	reduced	none
pre-presbyopic	myope	yes	normal	hard
pre-presbyopic	hypermetrope	no	reduced	none
pre-presbyopic	hypermetrope	no	normal	soft
pre-presbyopic	hypermetrope	yes	reduced	none
pre-presbyopic	hypermetrope	yes	normal	none
presbyopic	myope	no	reduced	none
presbyopic	myope	no	normal	none
presbyopic	myope	yes	reduced	none
presbyopic	myope	yes	normal	hard
presbyopic	hypermetrope	no	reduced	none
presbyopic	hypermetrope	no	normal	soft
presbyopic	hypermetrope	yes	reduced	none
presbyopic	hypermetrope	yes	normal	none

Find the data on the contact lens in Table 1.1. This gives the circumstances under which an optician will choose to recommend soft contact lenses, hard contact lenses, or no contact lenses at all; we'll talk more about what will feature means later. Every table line is one of those examples. This knowledge may be part of a structural classification as follows:

```
If tear production rate = reduced then recommendation = none
Otherwise, if age = young and astigmatic = no
                then recommendation = soft
```

Structural definitions do not always need to be couched like those laws. Other common means of expression are the Decision trees, which define the sequences of decisions to be made and the resulting recommendation.

It is a rather simplistic example. Next, the table represents all combinations of potential values. There are 24 rows reflecting three potential age values and two values for spectacle prescription, astigmatism, and tear levels of development (3 x 2 x 2 x 2 = 24). The rules don't really generalize from the data; they simply sum it up. The set of examples given as feedback in most learning situations is far from complete, and part of the task is to generalize to other, new examples. You can imagine omitting some of the rows in the table for which the production rate of tears is reduced and which still has the law.

If the tear production rate = reduced then recommendation = none

That will generalize to and fill in the missing rows correctly. Second, all of the examples define values for all of the functions. Real-life datasets often contain instances in which, for some cause or other, the values of such features are unknown — for example, measurements were not taken or lost. Third, the preceding rules correctly classify the instances, while sometimes misclassifications occur even on the data used to train the classifier, due to errors or noise in the data.

Machine Learning

Now that we have knowledge of the inputs and outputs, let's turn to learn machines. Anyway, what is learning? What is a thinking machine? Those are philosophical questions, and in this book, we are not going to be much concerned with philosophy; our focus is squarely on the practical. In the beginning, however, it's worth spending a few moments on basic problems, just to see how difficult they are, before rolling up our sleeves and digging at realistic machine learning. Our dictionary describes how to "learn":

Having information through research, practice, or teaching;

To become conscious by knowledge or by observation;

- Committing to Memory;
- Ascertain, to be aware of;
- To get instruction.

When it comes to thinking about machines, certain concepts have certain shortcomings. For the first two, checking whether or not the learning has been completed is nearly impossible. How do you know that a computer knows something? You would not only be able to ask them questions; even though you did, you would not be testing their ability to understand but would be testing their ability to answer questions. How do you know if anything has been aware of that? Whether computers can be awake, or aware, is a burning philosophical question. As for the last three words, while we can see what they signify in human terms, it seems that pure "committing to memory" and "receiving instruction" is far short of what machine learning would mean. They are too passive, and we know those activities are trivial to computers. Instead, in new contexts, we're interested in performance enhancements, or at least the performance potential. By rote learning, you can "commit something to memory" or "be aware of something," without being able to apply the new information to new circumstances. You can get guidance without having any value from it.

Previously we described data mining operationally as the process of automatically or semi-automatically discovering patterns in large quantities of data — and the patterns must be useful. For learning an organizational concept can be formulated the same way:
People learn as they change their actions in a way that will boost their success for the future.

It ties learning more to success than to awareness. You can check learning by evaluating and comparing the action to previous behavior. It is a description of a far more realistic kind and seems much more appropriate.

Yet there is still a problem here. Training is something of a slippery term. Some things change their behavior in ways that would make them do better in the future, but we wouldn't like to claim they've really learned. A good example is a slipper, which is convenient. Has it learned your foot shape? Surely it has changed its actions to make it as a slipper work better! And we would hardly like the learning to be named. Sometimes we use the term "reading" in everyday language to describe a kind of senseless learning. We train animals and even plants, but to think of training items like slippers that are not alive in any way would be to stretch the word a little. Yet it is special for writing. Training requires learning. Training implies a goal. Anything that knows has to consciously do so. That's why we wouldn't say a vine has learned to grow in a vineyard around a trellis — we'd say it's been conditioned. It is pure training to learn without intent. Or, more to the point, the learner's interest is in learning while the teachers are in teaching.

Thus on closer analysis, the second concept of learning has its own problems when it comes to thinking about computers in practical, performance-oriented terms. To determine if something has actually been learned, you need to see whether there was some intent involved or whether it wanted to. It makes the definition meaningless when applied to computers, as it is unknown if objects will behave intentionally. Philosophical discussions of what "learning" really means, like discussions of what "mission" or "meaning" really means, are fraught with difficulty. Also, the law courts find it hard to compete with motive.

Data Mining

Thankfully, the kind of learning strategies discussed in this book do not pose such conceptual problems — they are called machine learning without really taking any specific philosophical position on what learning actually is. Data mining is a practical subject and requires learning in a practical sense, not abstract. We are interested in techniques to identify and define structural trends in data as a method to help understand and make predictions from that data. The data may take the form of a series of examples — examples of clients, for example, that have exchanged loyalties, or cases where certain types of contact lenses might be required. The performance takes the form of predictions about new examples — a prediction of whether a particular customer is going to turn, or a prediction of what kind of lens is going to be recommended under specified conditions. However, since this book is about identifying and explaining patterns in the data, an explicit explanation of a system which can be used to identify unknown examples to illustrate the decision may also be included. As well as efficiency, the provision of a clear representation of the acquired information is helpful. Essentially, this represents all commonly considered concepts of learning: gaining knowledge and being able to use it.

Most learning strategies search for structured explanations of what is learned, definitions that can become relatively abstract, and usually articulated assets of rules like those previously described or the decision trees later described in this chapter. Since people will understand them, these explanations help to illustrate what has been observed and to clarify the groundwork for future predictions. Experience demonstrates that the specific information constructs that are learned, the conceptual descriptions, are at least as important in many applications of machine learning to data mining and often much more important than the ability to perform well on new examples. People also use data mining to gain information and not just predictions. It clearly seems like a good idea to gain information from results, if you can. Read on to find out how!

Simple examples: Weather and other problems

Within this book, we use a lot of examples, which seems especially fitting because the book is all about learning from examples! There are several regular datasets that we will frequently come back to. Different datasets continue to present new issues and challenges, and when evaluating learning approaches, it is useful and instructive to consider a number of problems. In addition, the need to work with different datasets is so significant that a corpus containing about 100 examples of problems has been put together to evaluate and compare various algorithms on the same set of problems.

The diagrams here are all unrealistically simple. Serious application of data mining includes thousands, hundreds of thousands of individual instances, or even millions. But when we explain what algorithms do and how they work, we need clear examples that clearly capture the essence of the problem but are small enough to be understandable in every detail. In the novel, we will deal with the illustrations in this section, and they are intended to be like "academic," in the sense that they will help us understand what is going on.

Another problem with the actual databases in real life is that they are mostly proprietary. This one will share with you their consumer and product selection information, and you can understand the specifics of their data-mining program and how it operates. Corporate data is a valuable commodity, one whose importance in the creation of data mining techniques such as those discussed in this book has increased enormously. However, here we are concerned with understanding how data mining methods work and getting the details of these methods so that we can able to trace their operation on actual data. Therefore our illustrations are clear. But they aren't simplistic: they represent the properties of real datasets.

The Weather Question

The weather problem is a tiny dataset we can regularly use to demonstrate the methods of machine learning. Fully fictional, it allegedly concerns the conditions that are necessary for playing some unspecified game. In general, instances in a dataset are defined by values of characteristics, or attributes, which quantify various aspects of the instance. There are four characteristics in this case: outlook, temperature, humidity, and windy conditions. The end result is whether to play or not.

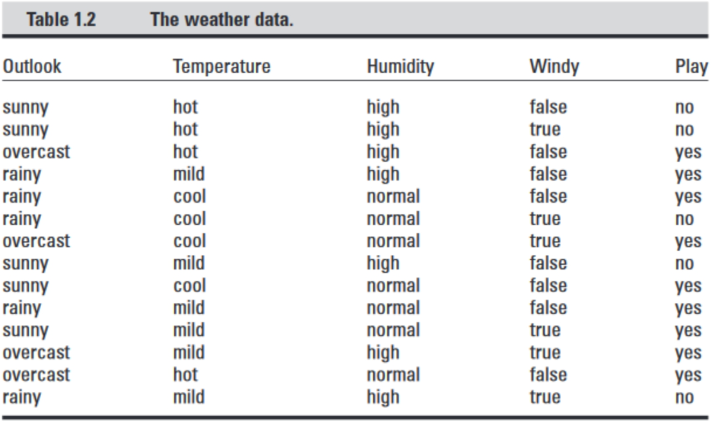
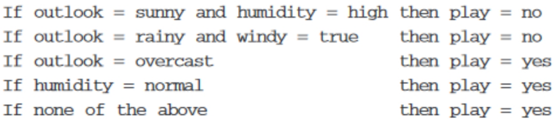


Table 1.2 The weather data.

Outlook	Temperature	Humidity	Windy	Play
sunny	hot	high	false	no
sunny	hot	high	true	no
overcast	hot	high	false	yes
rainy	mild	high	false	yes
rainy	cool	normal	false	yes
rainy	cool	normal	true	no
overcast	cool	normal	true	yes
sunny	mild	high	false	no
sunny	cool	normal	false	yes
rainy	mild	normal	false	yes
sunny	mild	normal	true	yes
overcast	mild	high	true	yes
overcast	hot	normal	false	yes
rainy	mild	high	true	no



```
If outlook = sunny and humidity = high then play = no
If outlook = rainy and windy = true     then play = no
If outlook = overcast                   then play = yes
If humidity = normal                    then play = yes
If none of the above                    then play = yes
```

In its simplest form, which is shown in Table 1.2, all four attributes have values that are not numbers but symbolic categories. The outlook may be sunny, overcast, or rainy; temperature may be hot, mild, or cool; humidity may be high, or normal, and windy may be real, or wrong. This generates 36 possible combinations (3 x 3 x 2 x 2 = 36), of which 14 occur in the set of examples data.

A collection of rules that have been learned from this information

— not necessarily a very good one — might look like:

Such laws are supposed to be understood in order: first, then if the second does not apply, and so forth. A collection of rules arranged to be interpreted sequentially is called a list of decisions. Interpreted as a list of decisions, the rules correctly define all the examples in the table while some of the rules taken individually are incorrect out of context. For e.g., if humidity = normal then play = yes, the rule gets one of the wrong examples (check the one). The meaning of a set of laws depends on how they are interpreted — not surprisingly!

Two of the parameters-temperature and humidity-have numeric values in the somewhat more complicated form shown in Table 1.3. This implies that every learning process, as in the former case, would establish differences concerning such characteristics, rather than simple equality tests. It is considered a problem with the numeric attributes — in this case, a problem with the mixed attributes since not all attributes are numeric.

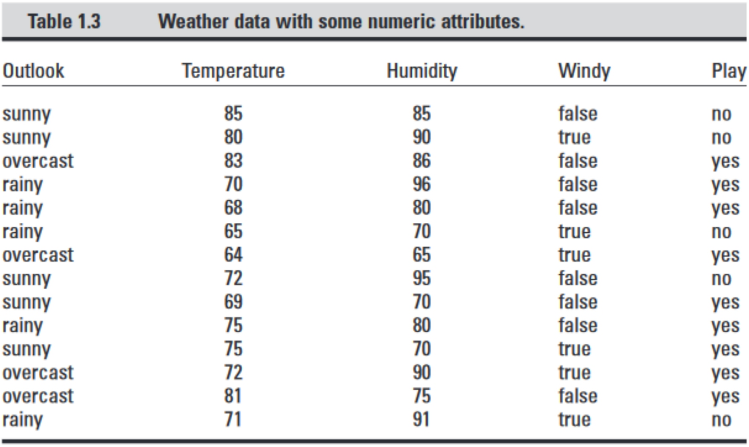


Table 1.3 Weather data with some numeric attributes.

Outlook	Temperature	Humidity	Windy	Play
sunny	85	85	false	no
sunny	80	90	true	no
overcast	83	86	false	yes
rainy	70	96	false	yes
rainy	68	80	false	yes
rainy	65	70	true	no
overcast	64	65	true	yes
sunny	72	95	false	no
sunny	69	70	false	yes
rainy	75	80	false	yes
sunny	75	70	true	yes
overcast	72	90	true	yes
overcast	81	75	false	yes
rainy	71	91	true	no

The first rule given earlier may now take the form of:

If outlook = humidity and sunny > 83 then play = no

It takes a much more complicated method to come up with rules involving numerical checks.

The rules we've seen up to now are rules of classification: they forecast example classification as to whether to play or not. It is equally possible to disregard the description and simply search for some laws that strongly equate different values of attributes. They are called the Laws of Association. Several association rules can be obtained from the data in Table 1.2 on the environment. The following are some good ones:

All these rules on the given data are 100 percent correct; they do not allow false predictions. The first two are applicable to four cases in the dataset, the third to three cases and the fourth to two. There are several other rules: in addition, it can be found that almost 60 association rules apply to three or more examples of the weather data and are absolutely accurate on this data. If you're searching for rules that are less than 100 percent right, you'll find a lot more. There are in numbers because, unlike classification rules, the rules of association can "predict" all of the attributes, not just a given class, and can even predict more than one. For example, the fourth rule predicts both a sunny outlook and high humidity.

4.1 Machine Learning And Statistics

What's the main difference between machine learning and statistics? Cynics, wryly looking at the explosion of commercial (and hype) interest in this field, equate data mining with statistics plus marketing. In addition, you shouldn't be searching for a dividing line between machine learning and statistics because there is a continuum — and a multidimensional one at that — of techniques for data analysis. Others originate from the skills learned in traditional statistics classes, whereas others are more closely aligned with the type of computer science machine learning that has emerged. The two sides have had very different cultures, historically. When forced to point to a single focus distinction, it may be that statistics were more concerned with testing hypotheses, while machine learning was more concerned with formulating the generalization process as a quest for potential hypotheses. Yet this is a gross oversimplification: statistics are much more than hypothesis testing, and many techniques of machine learning require no searching at all.

In the past, machine learning and statistics have developed very similar approaches in parallel. One is an induction by the decision tree. In the mid-1980s, four statisticians (Breiman et al . 1984) published a book on Classification and Regression Trees, and a popular machine learning researcher, J, in the 1970s and early 1980s. Ross Quinlan developed a method for inferring from examples to classification trees. Such two separate projects developed very similar methods for producing trees from examples, and only much later did the researchers become aware of each other's work. A second field where similar approaches emerged includes the use of nearest-neighbor detection methods. Those are basic statistical techniques that were widely adopted by researchers in the field of machine learning, both to improve classification efficiency and to make the process more computationally effective.

Yet now the two outlooks intersect. Within this book, the methods we will discuss include a lot of mathematical analysis. At the outset, traditional statistical methods apply when building and refining the initial example set: data visualization, selection of attributes, discarding outliers, and so on. Most learning algorithms use statistical tests in the creation of rules or trees and in the correction of "overfitted" models because they rely too heavily on the specifics of the specific examples used to generate them (we have already seen an example of this in the two decision trees in Figure 1.3 for the question of labor negotiations). The statistical tests are used to validate models of machine learning and to evaluate algorithms of machine learning. We'll learn a great deal about statistics in our study of realistic techniques for data mining.

Generalization As Search

Another way to picture the learning problem – and one that separates it from mathematical solutions – is to imagine a search for one that matches the data across a space of potential idea descriptions. Although the idea of generalization as a quest is a powerful philosophical resource for thinking about machine learning, understanding the practical approaches discussed in this book is not necessary. That is why this segment is potentially labeled, as shown by the margin of the gray bar.

Suppose concepts — the product of learning — are represented as rules such as those provided in Section 1.2 for the weather problem (although other concept representation languages can do the same). Suppose we list all possible sets of rules and then look for ones that follow a set of examples. Good work? Hey. An endless job? It seems so at first glance because there is no limit to the number of laws that might be there. Yet the number of potential sets of laws literally is finite. Note that first of each individual rule is no greater than a specified maximum size, with a limit of one term for each attribute: this includes four terms in total for the weather data from Table 1.2. As the number of possible rules is finite, the number of possible sets of rules is also finite, albeit incredibly high. But we would hardly be interested in sets containing a very large number of laws. In addition, we would hardly be interested in sets that had more rules than examples, because it's hard to imagine having more than one rule per example. And if we were to limit consideration to sets of rules smaller than that, the issue would be greatly reduced, but still very high.

For the second variant of the weather problem in Table 1.3, the possibility of an infinite number of potential concept definitions seems more severe, as such rules include numbers. If they are real numbers, you cannot even, in theory, enumerate them. Nevertheless, the problem disappears again on reflection since the numbers actually just represent breakpoints in the numeric values that appear in the examples. Take the temperature attribute, for example, in Table 1.3. It contains the different numbers 64, 65, 68, 69, 70, 71, 72, 75, 80, 81, 83, and 85—12. There are 13 possible locations where we can want to set a breakpoint for a temperature-related law. The problem, after all, is not infinite.

Thus the generalization method can be viewed as a search through a large yet finite search space. The problem can be solved in theory by enumerating definitions and scratching out those that don't match the examples given. A positive example excludes all terms. It doesn't match, and a negative one eliminates all it matches. The collection of remaining definitions shrinks for each example (or stays the same). When only one remains, this is the definition of the target — the idea of the target.

Unless there are few definitions remaining, unknown objects can still be labeled with them. An unknown object which matches all the remaining descriptions that should be classified or listed as matching of the target; if it fails to match any description, it should be classified as outside of the target. There's confusion here even when it suits certain definitions but not others. In this case, if the classification of the unknown object was revealed, the set of remaining definitions would shrink as the rule sets that categorized the object would be rejected in the wrong way.

4.2 The Processing And Ethics Of Data Mining

Using data — especially data about people — for data mining has significant ethical consequences, and data mining techniques practitioners have to behave responsibly by being aware of the ethical concerns surrounding their specific application.

Data mining is also used to discriminate when applied to individuals-who get the loan, who gets the discount deal, and so on. Some types of discrimination — racial, ethnic, religious, etc.—are not only immoral but illegal. The situation, however, is complex: This all depends on the demand. Using sexual and racial information for medical diagnosis is definitely ethical, but using the same information when there is no activity in paying for mining loans. Even when confidential information is discarded, there is a possibility of constructing models that rely on variables that could be used to replace racial or sexual characteristics. For instance, people often live in places identified with specific ethnic identities, and using an area code in a data mining analysis runs the risk of creating race-based models — though racial information has been specifically omitted from the data.

It is generally agreed that before people make a decision to provide personal information they need to know about how it will be used and what it will be used for, what measures will be taken to protect its security and dignity, what are the consequences of sharing or withholding the information, and any rights of redress they might have. If such knowledge is collected, such items should be explained to individuals — not in small legalistic print but in plain language so they will understand.

The future use of data mining techniques means that the ways in which a data archive can be used can extend far beyond what was initially conceived when the data was collected. It presents a major problem: the circumstances under which the data was obtained must be decided, and for what plan or purposes it can be used. Does data ownership grant the right to use it in other ways than those claimed when it was first recorded? That specifically does not apply in the case of directly collect personal data. Yet the situation, in general, is a dynamic one.

Surprisingly, things come out of data mining. It has been stated, for example, that one of France's leading consumer groups has found that people with color of red cars are more likely to default on their car loans. How is it

Such "discovery" status? Upon what knowledge is that based? What were the requirements for collecting the information? What is it legal to use it in? Certainly, insurance firms are in the business of discriminating between individuals on the basis of stereotypes — young males pay heavily for car insurance — but these prejudices are not based purely on statistical correlations; they also include the experience of the environment as common sense. If the preceding result suggests something about the sort of person who prefers a red car, or whether it can be ignored as meaningless, is a matter for human judgment based on personal experience rather than merely statistical criteria.

4.3 Data Mining Applications

Within this segment, we have based some of the data mining applications, and their techniques are discussed in order, respectively.

Medical data processing software

Health data mining applications can have enormous potential and practicality. The effectiveness of data mining in healthcare, however, depends on the availability of clean data on healthcare. It is critical in this respect that the healthcare industry examine how data can be better captured, stored, prepared, and mined. Possible directions include standardizing clinical vocabulary and sharing data between organizations.

Boost the advantages of Data Mining technologies in healthcare

Data mining is used for the study of the business market

The data mining technique is used in the MBA(Market Basket Analysis). When the consumer decides to purchase any goods, then this technique helps us find the ties between the different things that the consumer puts in their shopping buckets. There is the identification of these partnerships that support the business strategy. In this way, the retailers use the data mining technique to determine what consumers are intensifying (purchasing the particular pattern). The business and also assist in buying related items.

Data mining uses emerging trends in the education system around the world. But the education systems are changed day by day, and the order of UGC establishes a huge number of universities in the 21st century. As the number of universities is set side by side, a millennium of students is enrolled across the country each and every day. With huge numbers of aspirants to higher education, we agree that data mining technology will help address knowledge gaps in higher education systems. The hidden patterns, associations, and anomalies that are discovered from educational data through data mining techniques can improve decision-making processes in higher education systems. This improvement will bring benefits such as improving the productivity of the educational program, decreasing the drop-out rate for the student, and raising the promotion rate for the student, raising the retention rate for the student, increasing the transfer rate for the student, increasing the educational progress ratio, increasing the success of the student, increasing the learning output for the student, and reducing the cost of system processes. In this present era, we use the KDD and the data mining techniques to derive the information that this information can be used to enhance the quality of education. In this form of application, decisions are used to identify trees.

Data mining is now being used in the development engineering industry in several different areas.

When we collect the data from the production system, the customer will use the data for various purposes, such as discovering the errors in the data, enhancing the design methodology, making the data good quality, how best to help the data to make the decision. Much of the time, however, the data can be analyzed first then after discovering the secret patterns that will regulate the production process that will further improve product quality. Because Software Importance

In the last 20 years, manufacturing mining has significantly increased; it is now necessary to study its history and application critically.

Data Mining software can be a generic application or a domain-specific application.

The data mining framework can be used for generic or domain-specific applications. Many generic data mining applications cannot make these decisions on their own but direct users for data collection, data mining method selection, and results in interpretation. The multi-agent-based data-mining program has the capability to apply automated data mining selection techniques. The multi-agent method is used at different levels[8]: first, to pose the best-suited decision to the consumer at the level of the concept hierarchy specification, then at the level of the outcome. This decision is stored in Knowledge Base for use in a subsequent decision-making process. Multi-Agent Framework Tool used for the development of generic data mining systems uses multiple agents to perform different tasks.
Database technology has developed from rudimentary file processing to evolving query- and transaction-based database management systems.

Data mining is the job of finding interesting trends from vast volumes of data where data can be saved in databases, data warehouses, or other repositories of information. It is somehow young interdisciplinary field, drawing from areas like database systems, data warehousing, statistics, machine learning, data visualization, information retrieval, and high-performance computing.

Pre-processing data is an important problem for data storage and data mining because real-world data appear to be unreliable, noisy, and inconsistent. Pre-processing data includes the cleaning of data, data integration, data transformation, and data reduction. Routines for data cleaning try to fill in missing values, smooth out noise while identifying outliers and correct data inconsistencies. Information integration integrates multi-source data to create a cohesive data store. Routines for data transformation convert the data into suitable forms of mining.

4.4 Frequent Patterns of Data Mining

However, the topic of regular pattern mining is strong. This unit is dedicated to the regular element set mining methods. We dig into the following questions: How do we locate regular itemsets from large amounts of data, where the data is either transactional or relational? Why do we mine multilevel and multidimensional interaction rules? Which rules of the association are most interesting? Where can the mining method be supported or directed to discover important connections or correlations? Where can we exploit user expectations or constraints to accelerate the mining process? The techniques learned in this unit can also be expanded as we will research in later units for more advanced ways of regular pattern mining, such as from sequential and organized data sets.

In this section, the basic algorithm for finding frequent itemsets that we look at how to generate strong association rules from frequent itemsets. This unit explains many variants for improved performance and scalability of the Apriori algorithm. It presents pattern-growth methods for frequent itemsets in mining, which restricts the subsequent search space to the data sets comprising the current frequent itemsets only. It introduces methods that take advantage of the vertical data format for the mining of regular element sets.

Finally, we're exploring how the sequence mining findings can be applied to a particular application domain. The goal of sequence mining is to discover a collection of attributes, shared over time between a large and big number of objects in a given database. Consider, for example, a bookstore's sales database, where the items represent customers, and the attributes reflect writers or books. Let's assume the database tracks any customer's purchased books over a period of time. The trends found are the sequences of books the customers most often buy. Consider another example at a common site of a web-accessible database, where an object is a web user, and an element is a web page. The patterns discovered are sequences of pages that are most frequently accessed at that site. This type of information or knowlegde can be used to restructure the website or to insert relevant links dynamically into web pages based on patterns of user access.

Frequent patterns are patterns (such as item sets, subsequences, or substructures) that frequently appear in a set of data. A collection of objects, such as milk and bread, that often appear together in a transaction data set, for example, is a regular itemset. A subsequence, such as first buying a PC, then a digital camera, and then a memory card, where it often occurs in a shopping history database, is a (frequent) sequential pattern. A substructure may refer to various structural forms, such as sub-graphs, sub-trees, or sub-lattices, that can be combined with itemsets or sub-sequences. If a substructure frequently occurs, it is called a structured pattern (frequent). Finding these regular trends in mining interactions, similarities, and many other important data relationships play an essential role.

In addition, it also assists with data classification, clustering, and other data mining tasks. Also, pattern mining has thus become an important activity in data mining and a concentrated theme in data mining research. Within this section, we introduce the concepts of repeated patterns, comparisons, and similarities, exploring how they can be effectively mined.

A frequent pattern of mining searches in a given data set for recurrent connections. This section presents the basic principles of frequency pattern mining for database discovery. The earliest method of regular pattern mining for association rules by providing an example of market basket analysis. The basic principles of frequent patterns and associations in mining are providing a road map for the different forms of frequent patterns, association laws, and laws of connection that can be exploited. Interesting similarities and connections between itemsets in transactional and relational

Market Basket Analysis: A Motivating Example

Frequent itemset mining in large transactional or relational data sets leads to the discovery of associations and correlations among items. Despite vast quantities of data constantly being collected and processed, many companies are becoming involved in extracting these trends from their databases. For many business decision-making processes, such as catalog design, cross-marketing, and consumer shopping behavior analysis, the discovery of interesting correlation relationships within huge amounts of business transaction records will help.

Market basket analysis is one typical example of frequent itemset mining. This process analyzes the buying habits of customers by finding associations between the various items that customers place in their baskets. The discovery of such associations can help to develop marketing strategies by gaining insight into what items customers often buy together. For example, if consumers buy milk, how likely are they to buy bread (and what sort of bread) on the equal or same trip to the supermarket as well? This knowledge will contribute to increased sales by helping retailers do selective promotions and manage their shelf space.

Market basket analysis: Suppose as manager of a branch of All Electronics, you want to know more about your customers ' purchasing habits. Specifically, you ask, – What groups or collections of products are likely to attract customers on a specific trip to the store? · Business basket research can be conducted on retail consumer transaction data at your store to answer your query. The findings may then be used to prepare marketing or promotional campaigns or to develop a new catalog. For instance, market basket analysis can help you design different store layouts. In one strategy, items that are often purchased together can be placed in close proximity to further encourage the sale of those items together. If consumers who purchase computers often want to purchase antivirus software at the same time, placing the hardware display near the software display can help to increase both items' sales—placing hardware and software at opposite ends of the store in an alternative strategy which draws customers who buy these products to pick up other products along the way. For example, a customer may observe security systems for sale after deciding on an expensive computer while heading towards the software display to purchase antivirus software, and may also decide to purchase a home security system. Evaluation of market baskets will also help retailers prepare which products can be placed on sale at reduced prices. When consumers choose to purchase both computers and printers, getting a discount on printers will promote the selling of both printers and computers.

When we think of the universe as the set of items available at the store, then each object will have a Boolean variable reflecting the item's presence or absence. Then a Boolean vector of the values assigned to such variables will represent every basket. For purchasing trends that represent products that are frequently associated or purchased together, the Boolean vectors can be analyzed. Such trends can be expressed in the form of rules about associations. Since

For example, information which customers buying computers also prefer to buy antivirus software at the same time is shown in Association Rule (5.1) below:

Computer antivirus program [support = 2%; trust = 60%]

Law support and confidence are two indicators of interesting law. They represent the usefulness and certainty of rules discovered, respectively. Association rule support of 2% (5.1) means that 2% of all transactions under review indicate that machine and antivirus software are purchased together. A 60 percent trust means 60 percent of customers who bought a computer also purchased the app. Association rules are usually considered relevant if they meet both a minimum support threshold and a minimum confidence threshold. Users or Domain Experts can set such thresholds. More analysis may be carried out to show important statistical associations.

Mining frequent trends, without generation of candidates

Mining recurrent trends in transaction databases, time-series databases, and many other forms of databases were popularly studied in data mining investigations. Many of the previous studies follow a generation-and-test approach to the Apriori-like candidate range. However, the generation of candidates set is still costly, especially when prolific patterns and/or long patterns occur. In this analysis, we discuss a novel frequent pattern tree (FP-tree) structure, which is an extended prefix-tree structure for storing compressed, crucial information on frequent patterns, and create an efficient FP-tree-based mining process, FP-growth, to mining the complete collection of frequent patterns by pattern fragment development. Mining efficiency is achieved by three techniques: (1) a large database is compressed into a highly compact, much smaller data structure that prevents costly, repetitive database scans; (2) our FP-tree-based mining system adopts a pattern fragment growth method to avoid the expensive generation of a large number of candidate sets; and (3) a partitioning-based, dividing-and-conquer approach is used to decompose. Our performance analysis shows that the FP-growth method is effective and scalable for both long and short frequency patterns of mining, and is about an order of magnitude faster than the Apriori algorithm and faster than some other frequent pattern mining methods recently published.

R - Frequent Pattern Mining - FP-Growth Algorithm for Data Mining Algorithms:

The role of identifying recurrent trends in large databases is very significant in Data Mining and has been studied on a large scale in the last few years. Unfortunately, this task is computationally expensive, especially when there are a large number of patterns.
The FP-Growth Algorithm, proposed by Han, is an effective and scalable method for mining the full collection of frequent patterns by pattern fragment growth, using an expanded prefix-tree structure to store compressed and crucial information about frequent patterns called frequent-pattern tree (FP-tree). Han proved in his analysis that his method outperforms other common methods of frequent pattern mining, such as the Apriori Algorithm and the Tree Projection.

Frequent pattern: a pattern that appears regularly in a data series (a series of objects, subsequences, substructures, etc.);

- Motivation: The detection of inherent regularities in data
- What goods have also been purchased jointly? -- Diapers and beer?!
- Upon buying a PC, what are the following purchases?
- What kinds of DNA would this new product be immune to?
- Could we identify Internet documents automatically?

Practices

Basket data analysis, cross-marketing, catalog design, sale campaign analysis, web log analysis (click stream) and sequence analysis of the DNA

Why does Regular Pattern Mining matter?

- Reveals data sets' intrinsic and important property

- Forms the basis for various basic data mining activities

- Study of interaction, correlation, and causation

- Sequential, structural patterns (e.g., subgraphs)

- Pattern review in spatiotemporal, graphical, time-series, and stream data

- Grading: associative classification

- Analysis of the clusters: regular pattern-based clustering

- Warehousing data: cube iceberg and cube gradient

- The compression of semantic data: fascicles

Association Rules of Mining: A Recent Overview

Association rule of mining is one of the main and well-researched data mining techniques. This aims to extract interesting similarities, repeated trends, associations, or causal structures in the transaction databases or other data repositories among sets of objects. Rules of the association are commonly used in various fields such as telecommunications networks, competition and risk management, inventory control, etc. Different association mining techniques and algorithms are introduced briefly and compared later.

Association rule mining is to find association rules that satisfy a given database with predefined minimum support and confidence. Usually, the problem is broken down into two sub-problems. One is to find or explore those itemsets whose occurrences in the database surpass a predefined threshold; such itemsets are called regular or large itemsets. The second problem is generating association rules with the constraints of minimal trust from those broad itemsets. Suppose one of the major itemsets is Lk, Lk = {I1, I2, ..., Ik}, association rules are created with these itemsets as follows: the first rule is {I1, I2, ..., Ik-1} {Ik}, this rule can be determined as interesting or not by testing the trust. Then another rule is created by removing the last things in the precedent and adding it to the consequent, further testing the confidences of the new rules to decide their interestingness. Both processes are iterated before emptying of the antecedent. Since the second sub-problem is very straight forward, the first sub-problem is the subject of most studies.

The first sub-problem can further divided into two more sub-problems: the process of generation of large candidate itemsets and the process of generation of regular itemsets. We name those itemsets whose support exceeds the threshold of support as a broad or frequent object. The algorithms produce an incredibly large number of association rules in certain instances, sometimes in thousands or even millions. In addition, the rules of the association are often very specific. Comprehending or validating such a large number of complex association rules is virtually impossible for end-users, thus reducing the utility of the data mining performance. Many methods have been proposed to minimize the number of association rules, such as generating only — interesting rules, generating only — non-redundant rules, or generating only those rules that meet certain requirements such as scope, power, lift, or strength.

Rules of Association

Apriori is an algorithm for learning transactional databases over regular item set mining and the association law. This continues by defining the regular individual items in the database and expanding them to larger and with bigger item sets, as long as those item sets may appear in the database regularly enough. Apriori's regular itemsets can be used to evaluate association rules that illustrate general database trends: this has applications in domains such as business basket research.

Apriori is designed to work on transaction-containing databases (e.g., sets of items purchased by consumers, or Website frequency details). Certain algorithms are designed to find rules of association in data without transactions (Winepi and Minepi) or without timestamps (DNA sequencing). Every transaction is regarded as an itemset (an itemset). The Apriori algorithm determines the element sets that are subsets of at least transactions in the database, given a threshold.

Apriori uses a "bottom-up" approach that extends frequent subsets one item at a time (a step is known as candidate generation) and tests candidate groups against the data. When no further or more successful extensions are found, the algorithm ends.

Apriori uses a breadth-first search and a Hash tree structure to count sets of candidates.

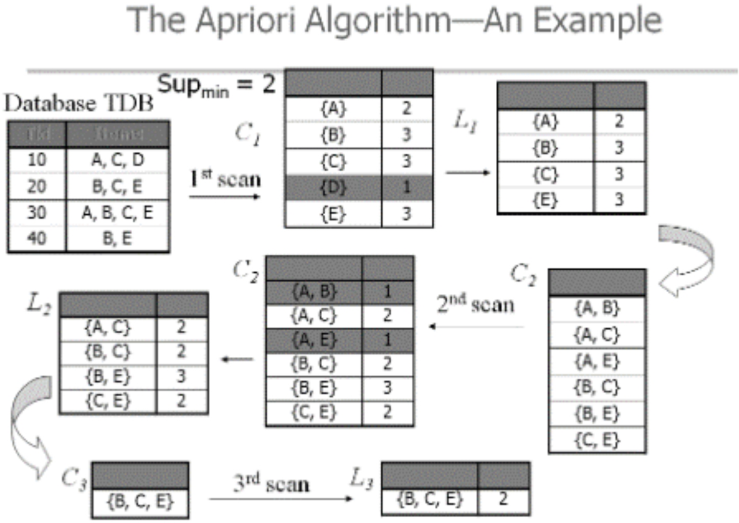
Fast. This produces nominee item length sets from object length sets. It then prunes candidates who have an uncommon subpattern. According to the downward lemma closure, all the regular -length element sets are in the candidate set. Subsequently, it scans the transaction database to determine frequent item sets between candidates.

Methodology:

Initially, search DB once to get 1-itemset frequently.

Generate length $(k+1)$ of candidate items from regular itemsets of length k Check the candidates against DB

Terminate when there is no frequency or candidate collection available



Database TDB

Tid	Items
10	A, C, D
20	B, C, E
30	A, B, C, E
40	B, E

C_1

{A}	2
{B}	3
{C}	3
{D}	1
{E}	3

L_1

{A}	2
{B}	3
{C}	3
{E}	3

C_2

{A, B}
{A, C}
{A, E}
{B, C}
{B, E}
{C, E}

C_2

{A, B}	1
{A, C}	2
{A, E}	1
{B, C}	2
{B, E}	3
{C, E}	2

L_2

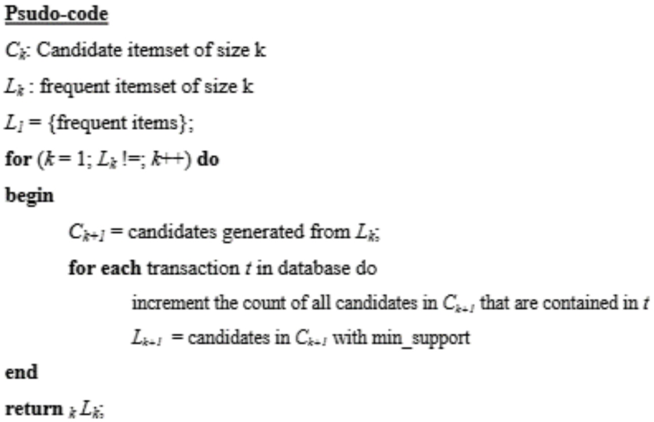
{A, C}	2
{B, C}	2
{B, E}	3
{C, E}	2

C_3

{B, C, E}

L_3

{B, C, E}	2



Psudo-code

```
$C_k$: Candidate itemset of size k
$L_k$ : frequent itemset of size k
$L_1$ = {frequent items};
for ($k$ = 1; $L_k$ !=; $k$++) do
begin
    $C_{k+1}$ = candidates generated from $L_k$;
    for each transaction $t$ in database do
        increment the count of all candidates in $C_{k+1}$ that are contained in $t$
        $L_{k+1}$ = candidates in $C_{k+1}$ with min_support
end
return $_k$ $L_k$;
```

The algorithm:

However, most methods use very complex internal data structures that have low locality and require additional space and overheads for computation.

SPADE is a new algorithm for fast Sequential Pattern discovery. SPADE uses combinatorial properties to break down the original problem into smaller sub-problems that can be solved independently in the main memory using effective lattice search techniques and easy joining operations. Both the sequences are contained in just three scans of the database. Experiments show that SPADE surpasses the best previous algorithm by a factor of two, and by a magnitude order with some pre-processed information. It also has linear scalability regarding the number of input-sequences and a number of other parameters of the database.

It's quite challenging to discover all the frequent sequences in large databases. The search space is incredibly big. For, eg, O.mk / potentially frequent sequences of length k occur with m attributes. With millions of items in the database, the I / O minimization issue is becoming paramount. Many existing algorithms, however, are iterative in nature, requiring as many full database scans as the longest frequent sequence, obviously a very costly operation. Some of the methods may be sensitive to the data-skew, particularly those using some form of sampling, which can adversely affect performance. Use Equivalence classes), to discover the set of all common sequences.

Through that database scans, SPADE not only minimizes I / O costs but also minimizes computational costs by using powerful search schemes. Even the approach based on the vertical id-list is insensitive to data skew. An extensive series of experiments show that if we have some extra offline knowledge, SPADE outperforms previous method or approaches by a factor of two, and by order of magnitude. In addition, SPADE scales linearly in the size of the database, and a variety of other parameters of the database.
Increasing the performance of algorithms on association rules

Mining association rules can reduce computational costs in four ways:

- Reducing the number of Server Passes

- By Server Sampling

- Apply additional constraints to pattern structure

- By parallelization.

There has been considerable development in all these areas in recent years.

Reduces the number of transits over the network

FP-Tree, regular pattern mining, is yet another landmark in the growth of association rule mining, breaking the Apriori's key bottlenecks. With just two passes over the database and without any phase of generation of candidates, the frequent itemsets are created. FP-tree is an extended prefix-tree structure that stores critical, quantitative pattern knowledge. Even regular length-1 objects would have nodes in the list, and the list nodes are arranged in such a way that nodes that occur more frequently will have greater chances of sharing nodes than those that occur less frequently.

FP-Tree scales even better than Apriori because as the help level decreases, the number, as well as the duration of regular itemsets, significantly increases. The candidate sets that Apriori must treat are incredibly broad, and it is very costly to suit the pattern of loads of candidates by looking through the transactions. The frequent pattern generation process includes two sub-processes: the FT-Tree construction, and the FP-Tree generation of frequent patterns. For Apriori sequence algorithms, the mining result is the same.

Decision trees are a classification technique in which the classification process is structured in a treelike structure using a collection of hierarchical decisions on the function variables. Usually, the decision at a particular tree node, referred to as the split criterion, is a condition on one or more function variables in the training data. The dividing standards split the training data into two parts or more. Consider the instance where Age is an attribute, for example, and the dividing criterion is Age 30. In this case, all training examples with age at most 30 are found in the left branch of the decision tree, while the right chapter includes all models with Age above 30. The aim is to identify a split criterion to minimize as much as possible the degree of "blending" of the class variables in each tree branch. Logically every node in the decision tree represents a subset of the data space defined in the nodes above by the combination of split criteria. Usually, the decision tree is built as a hierarchical partitioning of the training examples, just as a top-down clustering algorithm hierarchically partitions the data. The critical difference from clustering is that in training instances, the partitioning criterion is supervised in the class name's decision tree. Some classical algorithms for the tree of decisions include C4.5, ID3, and CART. An illustrative specimen will be used to demonstrate the basic idea of a decision tree construction.

The two function variables represent the attributes of Age and salary. Both parameters

Relates to the propensity to donate, which is also a class label. In particular, an individual's likelihood of donating is positively correlated with his / her Age and salary. But the best class separation can only be accomplished by combining the two attributes. In the decision tree construction process, the goal is to carry out a sequence of splits in a top-down fashion to create nodes at the level of the leaf where donors and non- donors are well separated. It illustrates a hierarchical arrangement within a treelike structure of the training examples. The split at first level uses the age attribute, while the division at the second level uses the salary attribute for both branches. Note that different splits do not need to be on the same quality at the same level of the decision tree.

The training specimen in all leaf nodes belong to the same class in this case, and thus there is no point in growing the decision tree beyond the leaf nodes. The splits indicated is known as univariate splits because they use one attribute. Using the split criteria to determine which branch to follow at each node of the tree, a single relevant track in the tree is traversed in a top-down fashion to classify a test instance. In the leaf node, the dominant class mark is identified as the appropriate class. For example, a test instance with Age below 50 and a salary below 60,000 can traverse the tree's leftmost path. Because this path's leaf node includes only non-donor training examples, the test instance will be marked as a nondonor.

In the split criteria, multivariate splits use more than one attribute. —— In this particular case, a single division leads to complete class separation. This implies multivariate requirements are better as they correspond to shallower trees. In general, shallower trees are more desirable for the same level of class separation in the training data because the leaf nodes contain more examples and thus are statistically less likely to overfit the noise in the training data.

A decision tree induction algorithm has two node forms, called the internal nodes and the leaf nodes. At that node, every leaf node shall be labeled with the dominant class. A unique internal node is the root node, which corresponds to the whole space of the function. The generic decision tree induction algorithm begins with the complete training data set at the root node and repetitively partitions it into lower-level nodes based on the dividing criterion. Only nodes containing a mixture of different groups need to be further broken down. The decision tree algorithm eventually stops the tree's growth, based on a stop criterion. The most straightforward stop criterion is one where all outstanding workouts in the leaf belong to the same class. One problem is that building up the decision tree to this level will result in overfitting, in which the model matches the training data's noisy nuances. Such a tree won't very well generalize to unknown test instances. To avoid the degradation of the accuracy associated with overfitting, the classifier uses a post pruning mechanism to remove excess nodes.

It is used for classifying unknown test instances with the use of top-down traversal from the root to a specific leaf after a decision tree has been created. For further traversal, the divided condition at each internal node is used to pick the correct branch of the decision tree. For the test instance, the label of the leaf node to be reached is reported.

4.5 FP Growth Algorithms

This section highlights the alternative methods available for frequent itemset generation. The explained Apriori Algorithm is one of the most commonly used algorithms in association mining but is not without its limitations. When there is a dense set of data, the performance of the Apriori Algorithm decreases, and another drawback is that it has a high overhead.

Even the way the transaction data set is represented will affect the algorithm's efficiency. The vertical data layout shown in the previous section examples is the more popular representation; the horizontal data layout is an alternative representation. The horizontal data plan can only be used for smaller data sets because the data set for the initial layout transaction may not fit in the main memory. Another crucial area that has improved or increased over the last couple of years is the traverse of the itemset lattice. The rest of this section adds alternative methods of generating frequent item sets, which take into account the aforementioned limitations and try to improve the Apriori algorithm's performance.

Mining Frequently Closed Sets

In this section, we've seen how frequent itemset mining can produce a huge number of frequent item sets, particularly when the min supply threshold is set low or long patterns exist in the data set. It showed that closed frequent itemsets9 could significantly reduce the number of patterns generated in frequent itemset mining while preserving full information about frequent itemset set. Which is we can easily extract the set of frequent item sets and their support from the set of closed frequency item sets. For a fact, therefore, it is more appropriate in most cases to mine the set of closed frequent item sets, rather than the collection of all frequent item sets.

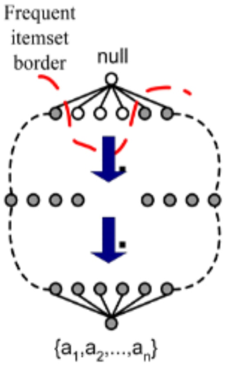
– How can we mine often closed item sets? · A naïve solution would be to mine the entire collection of frequent item sets first and then delete any frequent itemset that is a suitable subset of and bear the same support as the current frequent itemset. That, however, is very costly. This approach would first have to extract 2100 1 frequent item sets in order to obtain the frequent itemset length-100, all before it could begin to remove redundant item sets. This is prohibitively onerous. In addition, the data set contains only a very limited number of closed, frequent item sets

A suggested approach is to search directly during the mining cycle for closed frequency item sets. This allow or enable us to prune the search space as soon as the case of closed item sets can be found during mining. The tailoring techniques include:

Element merging: If each transaction containing a frequent itemset X also contains itemset Y but not a proper Y superset, then X [Y forms a frequently closed itemset, and there is no need to check for any itemset containing X but no Y., the predicted conditional prefix itemset database fI5:2 g is ffI2, I1g, fI2, I1, I3gg, from which we can see that each of its transactions contains itemset f Iggy Itemset fI2, I1 g can be combined with fI5 g to form the closed itemset, fI5, I2, I1:2 g, and we don't need to mine for closed I5 but not fI2, I1 g item sets.

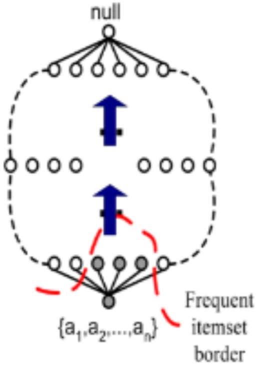
Traversal of Itemset Lattice Methods:

Generalized to Specific


Frequent itemset border
null
{a_1,a_2,...,a_n}

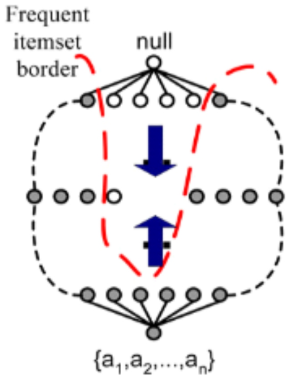
This is the approach which the Apriori algorithm employs. It uses frequent item sets beforehand to generate future item sets; by this, we mean it uses frequent k-1 item sets to generate k-itemset candidates. This technique is successful when a regular itemset's average duration isn't too long.

Specific to General


null
{a_1,a_2,...,a_n}
Frequent itemset border

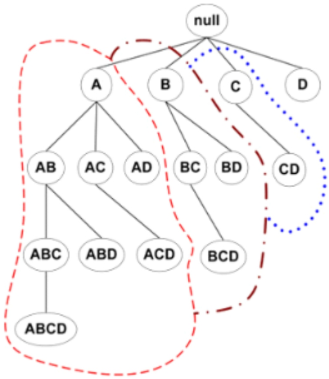
This technique is the general-to-specific reversal and does as the name implies. This begins with more common frequent item sets before the general frequency item sets are identified. This strategy helps us discover maximum frequent item sets in dense transactions where the frequent itemset boundary is near the bottom of the lattice.

Bi-directional



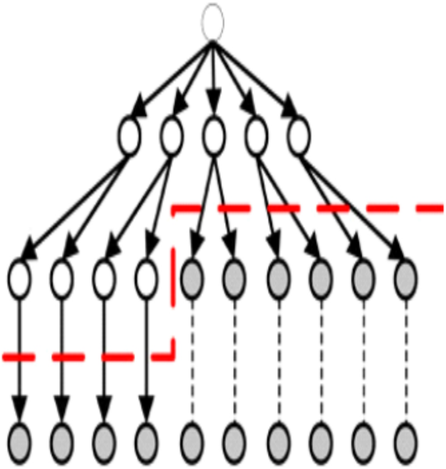
This technique is a variation of the previous two, and it's useful because it can easily define the regular border of item sets. It is highly productive when the regular boundary of the itemset is not located on either extreme, which is done conveniently by one of the previous strategies. The only drawback is that the storage of the candidate item sets requires more space.

Class Equivalence



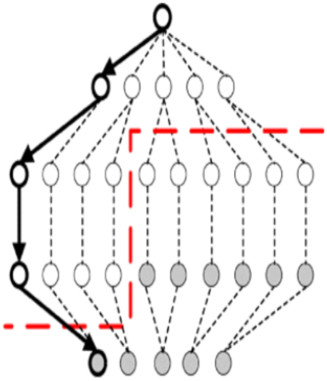
This strategy splits the lattice equivalence classes as seen in the figure and then employs a regular itemset generation algorithm, which thoroughly checks each class before moving to the next class. It is useful because it's understood that a certain itemset is a regular itemset. For example, if we knew that in most transactions, the first two items predominate, partitioning the lattice based on the prefix, as shown in the diagram, may prove beneficial, we can also partition the lattice-based on its suffix.

Breadth-first



This technique refers to the lattice as a breadth-first approach. This method first searches through the 1-itemset lines, discovers all the frequent 1-item sets to find frequent 2-item sets before going to the next level.

Depth First



This technique goes deep-first through the lattice. The algorithm takes a certain 1-itemset and follows it all the way down, until the 1-itemset as its prefix finds an infrequent itemset. This strategy is useful as it allows us to establish the distinction between regular and rare item sets quicker than using the breadth-first approach.

FP Algorithm on Growth

The FP-Growth Algorithm is another alternative way to find frequent item sets without the use of generations of candidates and thus improve efficiency. This is using a divide-and-conquer tactic for too much. The core of this method is the use of a special data structure called a frequent-pattern tree (FP-tree) that retains the information about the association of item sets.

This algorithm works as follows in simple words: first, it compresses the input database, creating an instance of the FP-tree to represent frequent items. After this first step, the compressed database is divided into a set of conditional databases, each associated with one frequently used pattern. Lastly, each such database is separately mined. Using this technique, the FP-Growth decreases the search costs by recursively looking for short patterns and then concatenating them into long regular patterns, providing good selectivity.

For major repositories, keeping the FP-tree in the main memory is not feasible. A technique for trading with this issue is first to partition the database into a collection of smaller databases (called projected databases), then to create an FP-tree from each of these smaller databases.

The following subsections explain the structure of the FP-tree and the FP-Growth Algorithm, and finally, an example is given to promote the comprehension of those concepts.

FP- Tree Structure

The frequent-pattern tree (FP-tree) is a compact structure that stores quantitative pattern information in a database.

Han describes the FP-tree as the structure of a tree described below:

- A root label with a set of item-prefix subtrees as children and a frequency-item-header table (presented in Figure 1 on the left side);

- Each node in the subtree with the item prefix consists of three fields:

- Item-name: registers representing which object is the node;

- Count: the number of transactions reflecting the portion of the route that enters the node;

- Node-link: refers to the next node in the FP-tree bearing the same item-name, or null where none exists.

- Each entry in the header table of frequencies consists of two fields:

- Item-name: equivalent to node;

- Node-link header: a pointer to the first node in the FP-tree carrying the element Name.

The frequency-item-header table may also have the support for an item count.

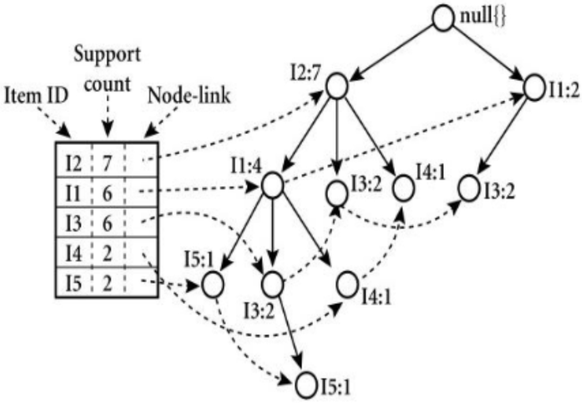

null{}
Support count
Item ID
Node-link
I2:7
I1:2
I2 7
I1 6
I3 6
I4 2
I5 2
I1:4
I3:2
I4:1
I3:2
I5:1
I3:2
I4:1
I5:1

Figure 6.1: An example of an FP-tree

The following Figure 6.1 provides an example of an FP-tree.

The original algorithm for Han-defined FP-Tree construction is shown below.

Algorithm 1: The creation of FP-trees

Input: A DB transaction database and a threshold for minimum support?

Output: DB 's frequent-pattern tree, FP-tree.

Method: The FP-tree is built like this.

- Scan the DB database transaction once. Collect F, the frequency range of items, and the support for each frequent item. Order F as FList, the list of regular items in support-descending order.

- Create the FP-tree root, T, and label it as zero. Trans in DB do the following for every transaction:

- Select and arrange the frequent objects in Trans according to the FList order. Let the list of sorted frequencies be [p P] in Trans, where p is the first element, and P is the rest. Call Tree Insert ([p P], T).

- The insert tree ([p P], T) function is executed as follows. If T has a child N such that N. item-name = p. item-name, then increase N's count by 1; otherwise build a new node N, its count initialized to 1, its parent linked to T, and its node-link connected to the nodes with the same item-name Via structure node-link. If P is nonempty, recursively call the insert tree (P, N).

The FP-tree is built in two database tests, using this algorithm. The first scan collects and sorts the collection of commonly used objects, and the second builds the FP-Tree.

Criterion for stoppage and pruning

The decision tree's stopping criterion for growth is intimately tied to the underlying pruning strategy. Suppose the decision tree is grown to the very end until each leaf node contains only instances belonging to a specific class. In that case, the resulting decision tree displays 100 percent accuracy on examples belonging to the training data. However, it often generalizes poorly to unseen test instances because the decision tree has now overfitted even in the training instances with random characteristics. The lower level nodes, which contain a smaller number of data points, contribute most of that noise. Simpler models (shallow decision trees) are usually superior to more complex models (deep decision trees) if they make the same error on the training data.

One possibility to reduce the amount of overfitting is to stop tree growth early. Unfortunately, the correct point at which to end the tree's growth is not known. Therefore, a natural strategy is to prune overfitting portions of the decision tree and convert internal nodes into leaf nodes. There are plenty of different criteria for determining whether a node should be pruned. One approach is to directly penalize model complexity by using the concept of minimum description length (MDL). In this approach, a tree's cost is defined by a weighted sum of its error (training data) and its complexity (e.g., node number). Tree complexity is calculated using information-theoretical concepts. So, the tree is designed to maximize the cost rather than just the error. The critical problem with this approach is that cost function is itself a heuristic one that does not fit well across different data sets consistently. An extra straightforward and intuitive approach is to hold out a small fraction of the training data (say 20 percent) and build the decision tree on the remaining data. On the holdout set, the impact of the pruning of a node on classification accuracy is tested. If the pruning increases the efficiency of the classification, then the node is pruned. Leaf nodes are iteratively pruned until the precision of the pruning can no longer be enhanced. Although such a method decreases the amount of training data for building the tree, pruning usually outweighs the tree-building process effect of training-data loss.

Practical Thematics

Decision trees are highly interpretable and straightforward to implement. Given sufficient training data, they can model arbitrarily complex decision boundaries. A univariate decision tree can model a complex decision boundary by constructing a sufficiently rooted tree with piecewise approximations. The main problem is that the amount of training data required to properly approximate a complex boundary with a treelike model is tremendous, and the dimensionality of the data increases. The resulting decision boundary is generally a very coarse approximation of the actual edge, with minimal training data. In such cases, overfitting is common. The decision tree's vulnerability exacerbates this problem to the division criteria at the tree's higher levels. A closely related classifier called rule-based classifiers can alleviate these effects by moving away from a decision tree's strictly hierarchical structure.

Classifiers based upon rules

Rule-based classifiers use a set of rules "if-then" = R1 ... Rm to match the resulting antecedents. Usually, a state is expressed in the form of:

IF Condition THEN Finally.

The condition on the left edge of the rule also referred to as the precedent, may contain a variety of logical operators, such as <, >, =, or applied to the variables of the function. The rule's right side is called the consequent, and Includes the class variable. Therefore, a Ri rule is of the Qi type, where Qi is the precedent, and c is the class component. The symbol "C" denotes the condition "THEN." During the training period, the rules are generated from training data. Qi Notation Represents a precondition for the set of features. In specific classifiers, such as association pattern classifiers, the prerequisite in the feature space corresponds to a pattern, but this may not always be the case. The requirement, in general, can be any arbitrary condition.

On variables in the function. Then these rules are used for classification of a test case. A state is said to cover a training instance when the training instance matches the condition in its antecedent.

An option tree can be seen as a particular case of a rule-based classifier, where each decision tree path corresponds to a rule. The decision tree, for instance, in Fig. 10.3a Complies with the following set of rules:
Age / earnings = 50 AND wage = 60, 000
Age > 50 YOUR Salaries > 60, 000 Donors
Age > 50 years of age and salary = 50, 000
Age > 50 years of age and salaries > 50, 000

The logical phrase on the left is represented in conjunctive form, with a collection of logical operators "AND." Every of the other primitive conditions (such as Age 50) is called a conjunct. The rule set from a set of training data is not exclusive and depends on the

particular algorithm in hand. For example, the decision tree in Fig will generate just two rules.
50 years + 50 years of age + 50 years of age, 000 [2]
Age/50 + salary/50, 000 > 2 respectively

As with decision trees, succinct rules are generally more desirable, regarding the cardinality of the ruleset and the number of conjuncts in each standard. This is because such practices are less likely to overfit the data and make unseen test instances generalize. Note that the left-hand antecedents always correspond to a condition of the rule. Logical operators such as" "are implicit in many rule-based classifiers, such as association-pattern classifiers, and are omitted from the rule antecedent definition. For example, the case where age and salary are discrete into categorical values of attributes is considered.
Age [50: 60], wages [50, 000: 60, 000]

In such a case, the discrete attributes for age and salary will be interpreted as "objects," On the left side, an association pattern-mining algorithm will discover the itemset. In other law, the operator" "is implied. In this section, the Associative classifiers will be addressed in detail later.

A rule-based algorithm generates a set of guidelines for the training process. The test instance classification process discovers all rules which the test instance triggers. The test instance is said to cause a state when the test instance meets the antecedent's logical condition. In some cases, the check instance causes rules with different consequent values. In these instances, the class mark prediction includes methods to overcome the conflicts. Rulesets can satisfy one or more of the properties below.

Mutually exclusive rules : Each law covers a disjoint data partition. So, a test instance will cause at most one law. The property is satisfied by the rules created by a decision tree. However, if the regulations extracted are subsequently amended to reduce overfitting (as in some classifiers such as C4.5rules), the resulting provisions may not remain mutually exclusive.

Exhaustive rules: At least one law encompasses all of the data space. Each test instance then triggers at least one state. Also, this property satisfies the rules generated from a decision tree. Creating an exhaustive rule set is typically simple. It establishes a single catch-all standard, which ultimately includes the dominant class in the portion of the training data not protected by other rules.

Carrying out the classification is relatively straightforward when a set of rules satisfies all the properties mentioned above. The explanation for this is that each test instance maps to exactly one state, and multiple rules do not generate contradictions in class predictions. In cases where rule sets are not mutually exclusive, conflicts can be resolved in two ways within the rules triggered by a test instance:

Rule ordering: The rules are ordered by priority, which can be specified in various ways. One possibility is to use a rule-of-order quality measure. Some standard classification algorithms, such as C4.5 rules and RIPPER, use class-based ordering that prioritizes rules with a particular class over the other. The resulting set of laws in order is often referred to as a list of decisions. With an arbitrary test instance, the class label is identified as the appropriate one for the test instance due to the top triggered law. Some other rule which is activated is ignored. When no control is triggered, then a catch-all class is listed as the appropriate one by default.

Unordered rules : Rule ordering is not given priority. It is possible to report the dominant class mark amongst all the triggered rules. Such an approach may be more robust as it is not prone to a rule-ordering scheme's selection of the single law. Typically speaking, the training step is more effective as all rules can be extracted concurrently with pattern-mining techniques without thinking about relative ordering. Ordered rule-mining algorithms generally have to integrate rule-ordering with methods like sequential covering, which are computationally expensive, into the rule-generation process. On the other hand, the testing process of an unordered way can be more costly due to the need to equate a test instance to all the rules.

Why will the various rules be organized for the classification of test instances? The first option is to order the controls based on a quality criterion, such as rule confidence or weighted support and trust measure. The strategy, however, is seldom used. The rules are, in most cases, arranged by class. In some rare class applications, ordering all of the states that belong to the limited class first makes sense. RIPPER uses that kind of strategy. Individual classifiers, such as C4.5rules, use specific precision and information-theoretical criteria to assign courses.

The Rule of Decision Trees

As mentioned earlier in this section, rules in a decision tree can be derived from the different paths. C4.5rules, for example, derives rules from the decision tree C4.5. The series of split requirements refer to the antecedent of a corresponding rule on each direction of the decision tree. Therefore, at first sight, it seems that rule ordering is unnecessary because the rules generated are exhaustive and mutually exclusive. However, the rule-extraction cycle is followed by a rule-pruning step in which several conjuncts are pruned to remove overfitting from the law. Rules are processed one by one. In a separate holdout validation package, conjuncts are greedily pruned from them to increase the accuracy as much as possible on the covered instances. This approach is similar to the pruning of the decision tree except that one is no longer confined to pruning the conjuncts at the lower levels of the tree. Therefore, the pruning process is more robust than that of a decision tree, as it is not constrained by a tree structure underlying it. Duplicate laws may be the product of conjunct pruning. Such acts are done away with. The rule-pruning process expands the scope of the various rules, and thus, the mutually exclusive essence of the laws is lost. As a consequence, ordering the regulations is necessary once again.

All laws belonging to the class whose ruleset has the least length of definition are prioritized over other laws. A rule set's total definition length is a weighted sum of the number of bits required to encode the model's size (rule set), and the number of examples covered by the class-specific rule set in the training data belongs to another class. Generally, this strategy favors courses with a reduced number of training instances. A second solution is to order first-class whose set of laws has the least amount of false-positive errors on a different game of holdouts. In general, a rule-based version of a decision tree allows creating a more robust decision boundary with limited training data than the base tree from which the rules are created. It is partly due to the greater versatility in the model that is no longer limited by the straitjacket of an exhaustive and mutually exclusive set of laws. Consequently, the method is best applied to unknown test cases.

The first type of classifier is called a generative classifier, while the second is called a discriminative classifier. Both classifiers will be discussed in more depth in the following.

Classified Naive Bayes

For conditional probabilities, the Bayes classifier is based on the Bayes theorem. This theorem quantifies a random variable's conditional probability (class variable), provided obvious observations on the value of another set of haphazard variables (feature variables). The Bayes theorem is commonly used in mathematics and probability. Take the following example, based on Table 10.1, to understand the Bayes Theorem:

Example 10.5.1 A charitable agency seeks donations from individuals in popularity, 6/11 of which are over 50 years of Age. The organization has a success rate of 6/11 when soliciting donations, and the probability that the Age is larger than 50 is 5/6 of the individuals who donate. Given a person over 50 years old, what is the likelihood that he or she will grant?

Consider the case where event E matches (age > 50), and event D matches a person who is a donor. The goal is to assess the posterior probability P (D E). This quantity is mentioned as the "posterior" probability because it is dependent upon Event E-discovery that the person is over 50 years old. Before observing Age, the "prior" likelihood P (D) is 6/11. Knowledge of an individual's Age affects subsequent odds due to the apparent associations between Age and donor behavior.

Bayes theorem is helpful for estimating P (D E) because it is difficult to estimate P (D E) directly from the training data. After all, it is easier to estimate other conditional and prior probabilities such as P (E D), P (D), and P (E). Bayes Theorem clearly states the following:
P (D E) = P (E) (10.16) According to the Law,
P (E) (E)

Each one of the right-hand words is already recognized. The P (E)

value is 6/11, and the P (E D) value is 5/6. Also, prior probability P (D) before Age is known is 6/11.
Then the latter likelihood can be calculated as follows:
P = (D)
(5/6) (6/11)
= 5:6. (10.17) -(10.17)
Six/11

Therefore, if we had 1-dimensional training data holding only the Age, along with the class variable, this method might estimate the probabilities. Table 10.1 includes an example of training instances satisfying the requirements set out above. Table 10.1 also makes it possible to check that the percentage of individuals over a period of 50 who are donors is 5/6, which is in line with Bayes theorem. The Bayes theorem is not important in this particular case, since the classes can be predicted directly from an alone attribute of the training data. A query arises as to why the indirect use of the Bayes theorem is useful if, in the first place, the posterior likelihood P (D E) could be calculated directly from the training data. The explanation is that conditional event E typically corresponds to a combination of constraints on d variables with different characteristics, rather than one. It makes it much tougher to measure P (D E) directly, for example, the likelihood P (Donor Age > 50, Pay > 50, 000) is more difficult to estimate robustly from the training data as there are fewer instances in Table 10.1 that meet both the Age and pay conditions. With the rise in dimensionality, this problem increases. In general, it may be the instance that not even a single tuple in the training data meets all of these conditions for a d-dimensional test instance, with d conditions. The Bayes rule offers a way for P (donor age > 50, salary > 50, 000) to be expressed in terms of P (age > 50, salary > 50, 000). Through the use of a product-wise approximation known as the naive Bayes approximation, the latter is much easier to estimate, while the former does not.

4.6 Sequential Algorithms Cover

Sequential cover methods are frequently used to create ordered rule lists. Then, the classification process uses the top triggered rule to classify unseen test instances in this case. Sequential coverage algorithms such as AQ, CN2, and RIPPER are examples. The iterative sequential coverage method applies the following two steps to expand the rules from the training data set D until a stop criterion is met:

Learn-One-Rule : Choose a different class label and, as a consequence, define the "best" rule from the current training instances with this class label. Add that rule to the bottom of the list of commanded rules.

Prune training data: Remove the training instances in which the rule learned in the preceding phase applies. Any training instances that fit the regulation's precedent must be eliminated, whether the training instance class mark suits the result.
The above generic definition refers to all algorithms that cover a sequence. The various sequential algorithms that include primarily vary in the specifics of how the rules are organized concerning each other.

Class-based ordering: All rules corresponding to a particular class are generated and placed contiguously on the ordered list in most sequential covering algorithms such as RIPPER; Rare levels are typically ordered first. And groups that are put on the list earlier can be more preferred than others. Often this may trigger artificially lower precision for test instances belonging to the less favored class.

When class-based ordering is used, the rules are generated contiguously for a particular class. Adding standards for each type has a stop criterion, which is dependent on the algorithm. For instance, RIPPER uses an MDL criterion, which stops adding rules when further adding increases the model's description length by at least a predefined number of units. Another more basic stop criterion is when a preset threshold exceeds the next generated rule's error rate on a separate validation collection. Finally, one might use a limit as the class-specific stop criterion on the number of exposed training instances remaining for a class. When the number of exposed training instances left for a grade falls below a threshold, then rules for the following type will no longer be increased. At this stage, standards are being established, which correspond to the next level. The approach is repeated (k 1) times for a k-class problem. Rules for class kth are not being created. A single catch-all state is the least prioritized rule with its effect as the kth class. If the test instance does not fire rules that belong to the other courses, this class is considered the appropriate mark.

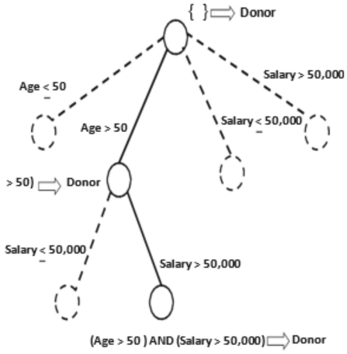
Quality-based ordering: The class-based order is not used in some covering algorithms. For selecting the next rule, a quality measure is used. For example, one may generate the state in the remaining training data with full confidence. Among the remaining test instances, the catch-all government corresponds to the dominant class. Throughout the practice, quality-based ordering is rarely used because of the difficulty throughout evaluating a quality criterion that is only established over the remaining test instances.

The Learn-One-Rule method will be defined under this assumption, as class-based ordering is more usual.

The Learn-One-Rule technique raises rules from the general to the specific, in much the same way a decision tree grows a tree from general nodes to specific nodes hierarchically. Note that a track in a decision tree is a law in which the antecedent corresponds to the combination of the split criteria at the different nodes, and the consequent corresponds to the leaf node name. When a decision tree grows several different disjoint paths at one time, a single "best" path is established by the Learn-One-Rule process. It is yet another example of the closeness of decision trees to rule-based approaches.

Learn-One-Rule 's concept is to add conjuncts successively to the rule's left-hand side to expand a single decision track (rather than a decision tree) based on a consistency criterion. The tree root conforms to rule c. Class c represents the consequence of the growing rule. In the simplest version of the method, by successively applying conjuncts to the antecedent, a single path is generated at one time. Conjuncts are used, in many words, to increase the consistency as much as possible. The rule's most straightforward consistency criterion is accuracy. The problem with this criterion is that regulations of high precision yet shallow coverage are not necessarily acceptable due to overfitting. The precise selection of the quality criterion regulating trade-off between accuracy and coverage will be discussed later in detail. As in the instance of a decision tree, it is important to check different logical conditions (or split choices) to decide the best conjunct to add. The method of enumerating the different split choices resembles a decision tree. The rule is established before an essential criterion of the stoppage is met. A natural measure of stoppage is when the quality of the government does not improve through further growth.

One matter with this technique is that it can lead to suboptimal rules if a mistake is made early during tree development. One way to reduce the probability of suboptimal practices is to follow the best m paths during rule-growth instead of a single one. For the donor case in Table 10.1, an example of rule development with the use of a unique decision path is shown in Fig. 10.5. 10.5. In this case, the donor class law is practiced. The added first conjunct is Age > 50, and the attached second conjunct is Salary > 50 000. Note the intuitive similitude between Figs' decision tree.



The consistency criterion for the growth of the paths during the Learn- One-Rule procedure remains to be defined. Under what basis is a particular path chosen over the others? The similarity between rule growth and decision trees suggests using similar measures, such as precision, entropy, or the Gini index, as used in decision trees for split criteria. The requirements need to be changed as a rule only applies to the training examples covered by the precedent and the consequent single class. At the same time, decision tree splits are evaluated at a given node and all categories concerning all training instances. Also, decision tree split measures need not take account of issues such as rule coverage. To stop overfitting, one would like to decide rules with broad coverage. For example, a jurisdiction

that only covers a single training instance will still have 100 percent accuracy, but unknown test instances typically don't generalize well. Therefore, one technique is to merge the requirements for accuracy and coverage into a single integrated approach.
Measurement.

The easiest approach to combination is to use Laplacian smoothing with β parameter

That regulates the smoothing level in a k-class collection of training data:

The parameter $\beta > 0$ governs the smoothing degree, $n+$ represents the number of correctly classified (positive) examples covered by the rule and $n-$ represents the number of (negative) incorrectly classified examples covered by the rule. Hence the total number of examples protected is $n+ + n-$. For cases where the absolute number of exams covered $n+ + n-$ is very high, Laplacian smoothing penalizes the accuracy to compensate for low coverage unreliability. The calculation thus supports greater reporting.

A second option is the statistical probability ratio. Let nj be the observed number of training data points covered by the rule of class j, and let be the expected number of covered examples that would belong to class j, if the class distribution of the covered examples is the same as the full training data. Or put it another way, if p1 ... Pk be a fraction of the examples in the complete training data belonging to each class, and then we have:
k
E= ii
(i=1)

Then the likelihood ratio statistic R may be calculated as follows for a k-class problem:
The value of R increases when class distribution in the covered examples is substantially different from that in the original training results. Thus, the statistics appear to favor covered standards whose distributions vary significantly from the initial training results. Also, raw frequencies are present n1 ... Nk as multiplicative factors of each of the words on the right side of the Eq. 10.14 Ensures a

broader scope of the rules

Recommended. The CN2 algorithm uses the scale.
A further criterion is the information gain from FOIL. The term "FOIL" is First Order.
Inductive Candidate. Take the case where the law protects $n+$
Examples are positive and $n-1$.

Negative examples where positive examples are defined as examples of training which match the
Class in accordingly. Consider the case where a conjunct is attached to the antecedent.
Adjust the number of instances positive and negative to $n+$ and $n-$, respectively.

Then the FG information gain from FOIL is defined as:
$fg = n2$
$\log 2\ n2 + n1 + n2\ -\log 2\ n1 + n1 + nt$

This measure tends to select rules with high coverage as $n+$ is a multiplicative factor in F G. At the same time, the information gain increases with greater accuracy due to the term inside the parentheses. The RIPPER algorithm uses a unique variable. As with decision trees, the rules can be widened until 100 percent accuracy is achieved on the training data, or when the added conjunct does not improve the rule's accuracy. RIPPER's criterion is that because of the inclusion of a conjunct, the total definition length of the law must not be extended by more than a given threshold. The length of a rule description is determined by a weighted function of the conjuncts' size and the examples misclassified.

Pruning rule

Rule-pruning is not only crucial for rules developed by the Learn-One-Rule method but also for techniques such as C4.5rules that extract the rules from a decision tree. Regardless of the method used to remove the law, too many conjuncts will result in overfitting. As with decision tree pruning, pruning can be achieved using the MDL theory. For example, one may add a penalty term π to the quality criterion in the rule-growth process for any conjunct in the law. This will result in a negative rate of error. Consequently, rules with more conjuncts would have more considerable aggregate penalties to compensate for their greater complexity in the model. A more straightforward approach to estimating pessimistic error rates is to use a different holdout validation package, which is used to measure the error rate (without a penalty) but is not used by Learn-One-Rule during rule generation.

The conjuncts that were added successively during the law (in sequential cover) are then checked in reverse order for pruning. When pruning decreases the negative rate of error in the training examples covered by the rule, then the generalized rule is used. Although some algorithms like RIPPER test initially introduced the most recent conjunct for rule pruning, doing so is not a strict requirement. To that the negative error rate as much as possible, it is possible to check the conjuncts for elimination in any order, or a greedy fashion. The pruning of rules will cause some of the states to become similar. This eliminates redundant controls from the rule collection before classification.

Classifying Relationships

Associative classifiers are a common strategy because they rely on pattern mining of associations, for which there are several effective algorithmic alternatives. For associative pattern mining algorithms. The discussion below assumes binary attributes, although any data type, as discussed in Chap, can be converted to binary attributes with the process of discretization and binarisation. Also, the rules generated by associative classifiers can be either ordered or unordered, depending on application-specific parameters, unlike sequential covering algorithms in which rules are always set. The critical feature of class-based association rules is that they are mined in the same manner as standard association rules, but in effect, they have a single class element. For an associative classifier, the basic strategy is as follows:

1. Measure all class-based association rules at a minimum level of support and trust.

2. Using the mined rules for classification, for a given test case.

There are a variety of choices for implementing the two steps. A naive way to enforce the first step would be to mine all the rules of association and then pick out only the rules under which the result applies to a class. However, such a method is inefficient as it produces several states with the consequence of nonclass. Also, the rule set includes considerable overlap, since many rules that have 100 percent confidence are individual cases of other regulations with 100 percent confidence. The pruning methods are, therefore, necessary during the process of rule-generation.

The Association-Based Classification (CBA) approach uses a variation of the Apriori method to establish associations that follow the respective constraints. Create 1-rule-items is the first step. These are newly created items corresponding to element combinations and class attributes combinations. Then, these rule items are expanded using conventional processing in the Apriori form. Another change is that when patterns are created that are associated with rules with 100 percent confidence, those rules are not expanded to keep the ruleset more general. Nearly any tree enumeration algorithm can use this broader approach. The bibliographic notes contain indications of several recent algorithms that use other frequent pattern mining methods to generate rules.

The second stage of associative classification is to make predictions for unseen test instances using the created rule collection. It can be used for both ordered or unordered strategies. The organized approach prioritizes the rules based on support (analogous to coverage), and confidence (similar to precision). Several heuristics can be used to establish an integrated ordering metric, using a weighted help and confidence combination. For discussion of a generic rule-based classifier, which uses various types of steps. The top m corresponding rules to the test instance are calculated after the rules have been ordered. For the test case, the dominant class label from the matching rules is identified as the relevant one. A second strategy does not request the rules, but from all the activated rules, it decides the dominant class name. Other heuristic strategies for the predictive process weigh the rules differently, depending on their support and confidence. Also, many variations of associative classifiers do not use the comfort or confidence to exploit the rules but use class-based dis-criminative methods for pattern mining directly. The comments in the bibliography contain examples of these approaches.

Probabilistic Categorizers

Probabilistic classifiers create a model that quantifies as a probability the relationship between the feature variables and the target (class) variable. Many ways are modeling like that can be achieved. Four of the most popular models are:

1. **Bayes classifier:** The Bayes law is used to model the likelihood for a given set of function variables of each value of the target variable. Similar to clustering mixture modelling, it is presumed that data points within a class are created from a particular distribution of probabilities, such as the Bernoulli multinomial distribution. Sometimes (but not always) a naive Bayes presumption of class-conditioned function independence is used to simplify modeling.

Chapter 5: Predictive Modelling of Data

Predictive modelling is one of the most prevalent tasks in data mining. As the name suggests, it is the process of collecting historical data (the past), identifying patterns in the data seen through some of the methodology (the model), and then using the model to predict what will likely to happen in the near future (scoring new data).

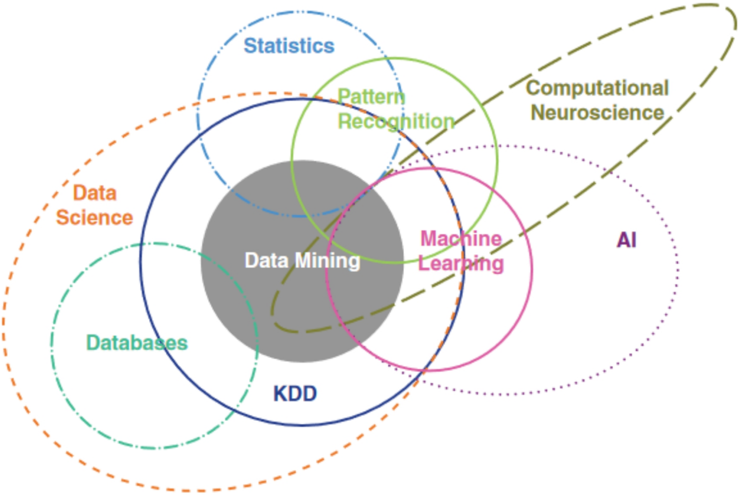


Figure 5.1 Multidisciplinary Nature of Data Mining

Data mining leads to composite discipline that overlaps other scientific branches. Figure 5.1 shows the contributions of many different fields to the development of data mining science. Because of the achievements of many fields, a continuous educational challenge is to stay up-to-date on the progress that being made in the field of data mining. Within this section, I address algorithms that come mainly from machine learning and statistics. These two groups largely live in different departments of the university (computer science and statistics respectively) and, in my point of view, are feuding about the best way to prepare students for the data science field. The Departments of Statistics teach a lot of theory but produce students with limited programming skills. Departments of computer science produce great programmers who have a good understanding of how programming languages communicate with computer hardware but have minimal training on how to interpret the data. This void demand that organizations train new hires in these fields. There is a recent trend to give students a more comprehensive education and better prepare them to make immediate and quick contributions to a role in data mining. Until these programs develop their student base, job applicants would probably only know half of the algorithms widely used for modeling. For statisticians, that is to say, regression, General Linear Models (GLMs), and trees for decisions. It's neural networks for the computer scientist, helping vector machines, and the Bayesian methods.

Predictive modeling isn't necessarily complicated or difficult; it's a fundamental task we learn as we grow and gain experience at an early age and hone. You do so naturally in your daily life multiple times a day. How long will your afternoon commute take, for example?
My most variable commute resided in the Washington, DC area. Those were the main factors in my commuting time:

- I left the office the time
- On the weekday
- Season (was its summer?)

If you work from home, there are probably common key variables in your commute, and maybe a couple more that were not relevant to my commute. Another forecast you make every day is what the weather will be like. My kids tend to wear shorts and T-shirts frequently later in the fall season than determines the social norm. This year, after the regular conversation about what the temperature would be during lunch, recess, PE, and the walk to and from school, my wife and I made a firm rule that pants and long-sleeve shirts were required after Veteran's Day (November 11). Glancing at the average degree of temperatures, it would have been reasonable to assume that the temperature would not rise to the point of "need" to wear shorts, but sometimes looking at averages is not a good technique for predicting. It was 75 degrees in Cary, North Carolina, in the first week of December, while it was at 30 degrees in Los Angeles.
This short story highlights a few key points concerning predictive modeling:

- Models are sometimes wrong.
- The farther the period of time, the more confusion there is.
- Averages (or averages) do not estimate extreme values.

George Box was a pioneer and influential statistician who taught at the University of Wisconsin (also the son- in- law of Sir R. A. Fischer). He published a book titled Empirical Model Building and Surfaces of Response in 1987. In it, he makes two statements related to the quality of predictive models now famous in the statistical community. He says: "Remember that all models are wrong; the practical question is how wrong they have to be in order not to be useful." Later in the same text, he says: "In essence, all models are wrong, but some are useful." These points are somehow important to bear in mind as you use increasing amounts of data to make more predictive. A model result should not ignore logic and reason. Following a methodology developed and using clear processes will help to reduce errors and enable you to improve processes when things go wrong.

Building on the weather analogy, it is clear that predicting something immediate for a stable process is more certain than farther out. Tomorrow's weather forecast is more likely to be accurate than the one in the future for five or ten days. The same applies to other predictable systems such as stock exchange, home prices in real estate, or gasoline. Statistically, this is referred to as an interval of prediction, and the farther time horizon you predict for becomes larger.

This chapter explains a methodology that was used by thousands and thousands of data miners to build predictive models. If the methodology is developed, knowledge about the types of predictive models and their commonality and distinction will be provided to you.

A Methodology for Generating Models

A lot of practitioners have developed and refined the process of building models over many years. This is a simple, proven approach to the development of successful and profitable models.

1. Prepare the numbers. This step must need to be completed before

any meaningful analysis or exploration can take place.

This is done in larger organizations by a separate team, and maybe even by a different business unit. No matter how your data is prepared, either through a formal request and accompanying specifications, as I did in the United States. This step is essential for a successful model if you have permission to query directly from the enterprise data warehouse (EDW).

Investment in understanding the process of preparing data within your organization often pays long term benefits. Broad knowledge of what data exists and how it can be combined with other data sources will provide insight that was not possible just a few years ago because you need access to increasingly larger and more granular data. When your IT department doesn't hold more data at longer and finer rates, then you're behind the curve and at risk of being obsolete in your business.

2. Perform exploratory analyzes of the data. This is the step in which you understand your data and begin to gain insight into the relationships among variables. If you don't have that expertise, this exploration is best done with a Domain expert. Using complicated exploration, fantastic data miners will discover relations and trends. They will present these findings to a domain expert, with great excitement, who will politely answer that those things have been known for some time. In today's business climate, you can't give a thorough and complete analysis and recommendation without domain expertise to complement analytical skills.

In recent years the tools used to analyze exploratory data have changed. This was done in the late 1990s using mostly tabular output, using SQL or a scripting language. There was to be pored over correlation matrix and static graphical output. The research was always sluggish and was a prerequisite for programming skills. Graphic resources have changed significantly over the last few years. In the space of business visualization, there are now many commercial products from SAS, IBM, and SAP, even smaller vendors as well such as QlikTech, and Tableau, to name a few. These products, generally through a browser-based interface, can load data for visual exploration and provide a highly interactive experience in data exploration. Its technology has been proven to operate with billions of observations, given there are ample hardware resources.

Software discovery never gets complete. There are often more ways to look at the data and experiences and relationships that need to be considered, so it is important to follow the theory of sufficiency and the law of lower returns. The law of decreasing returns comes from the field of economics and states that adding another unit of effort (time in our case) will yield less incremental value per unit for each successive unit of effort you put into the task. In this case, the insight and knowledge you gain about data exploration between hours 15 and 16 are most likely lower than the insight you gained between hours 2 and 3. The sufficiency theory accepts the rule of diminishing returns and sets a criterion for loss of efficiency. This is described in popular language: Knowing when to stop exploration. The methodology for software development has been moving to incorporate this idea through agile processes of learning a little, getting started, and continuous improvement.

3. Build your Model One. The secret to this move is to understand right up front that several iterations will require the effective model-building process. The Thomas Edison states during some ventures that "I haven't disappointed. I just found 10,000 ways that aren't going to work, "it's going to seem very apt. Until you build the first model, you won't be able to evaluate exactly what the model's potential impact will be. Build this model quickly with a method with which you are most at ease. (I often use a decision tree.) Building the first model helps to cement the success criteria and set the right expectations for those who will use the model predictions. Humans are to be optimistic in nature. We believe goods are going to sell more than they do, tasks are going to be done without problems, etc. Building the first model is a reality check on performance and expectations for the future. This first model is the Champion model by default.

4. Models are constructed to iteratively. This phase is where you should spend most of the time. This step is a feedback loop where you will build a model (the challenger) and then compare it to the model of the champion using some objective criteria which define the best model. If the challenger is better than the model of the champion, then assess if the model of the challenger meets the goals

of the project. If the goals of the project are not met, or the champion is not displaced, then construct another model. There's often no concrete model assessment to determine when to stop, but rather a time window that forces the project to end. Say that you are contracted for a marketing campaign to provide a list of customers. The campaign has a deadline to provide next Tuesday's customer list, so the model building will continue until that point in time.

sEMMA

sEMMA is a SAS data mining methodology that focuses on the logical organization of the data mining project model development phase. It describes the process of taking insight and knowledge from your data—the acronym sEMMA-sampling, exploring, modifying, modeling, evaluating-refers to the core process of data mining. Starting with a statistically representative sample of your data, sEMMA makes it easy to use exploratory visualization and statistical techniques, select and transform the most predictive variables of sig cant, model the variables to predict results, and confirm the accuracy of a model.

Let us address a common misconception before examining every phase of sEMMA. That confusion is referring to sEMMA as a technique for data mining. sEMMA is rather not a data mining methodology but rather a logical organization of SAS Enterprise Miner's functional toolset for performing core data mining tasks. Enterprise Miner can be used in some cases as part of any company that adopts iterative data mining methodology. It goes without saying about steps such as formulating a well-defined business or research problem and assembling representative data sources of quality are critical to the overall success of any data mining project. sEMMA focuses on the development of model aspects of data mining:

Sample your data (optional) by extracting a portion of a big data set. This has to be big enough to contain important information, yet small enough to manipulate fast. SAS Institute advocates a sampling strategy that applies a reliable, statistically representative sample of large fu detail data sources for optimum cost and performance. Mining a representative sample rather than the entire volume reduces the processing time needed to obtain crucial business information. If the data as a whole show general patterns, these will be traceable in a representative sample. If a niche is so tiny that it is not represented in a sample, and yet so important that it influences the big image, it can be discovered using summary methods. We also support the development of partitioned data sets using the node Data Partition:

- Training — used for fitting the pattern.
- Validation — used to evaluate and prevent overfitting.
- Testing — used to get an honest assessment of how well a model is generalizing.

To gain understanding and ideas, explore the data by searching for unanticipated patterns and anomalies. Exploration helps sharpen the process of discovery. If visual exploration does not show consistent patterns, statistical techniques like factor analysis, correspondence analysis, and clustering can somehow be used to explore the data. For example, clustering could reveal groups of customers with distinct ordering patterns in the data mining for a direct mail campaign. Knowing these trends provides incentives for personalized promotions or mailings.

Modify your data by selecting, creating, and transforming the variables in order to focus on the process of model selection. You may require to modify your data to include details, such as customer grouping and significant subgroups, or add new variables based on your findings during the exploration process. You would also need to search for outliers and lower the number of variables to narrow them down to the most important ones. You may need to adjust data as the "mined" data changes due to either new data being accessible, or new data errors found. Because data mining is an iterative process and dynamic, when new information is available, you can update the data mining methods or models.

Model your data by automatically allowing the software to search for a combination of data that reliably predicts a desired result. Data mining modelling techniques include neural networks, tree-based models, logistics models, and other mathematical models — such as the study of time series, memory-based reasoning, and key components. Each model type has different strengths and is suitable depending on the data within specific data mining situations.

Neural networks, for example, are very good at fitting highly complex nonlinear relations.

Test the data by measuring the usefulness and reliability of data mining results, and determine how well it performs. A common way of assessing a model is to apply it during the sampling phase to a portion of the data set aside. If the model is valid, then this reserved sample, as well as the sample used to build the model, should work. You can also test the model against known data similarly. If you know which customers, for example, you had high retention levels, and the model predicts retention, you should check to see if the model correctly selects those customers. Furthermore, practical model applications, such as partial mailings in a direct mail campaign, are helping to prove its validity.

By evaluating the results obtained from each stage of the sEMMA process, you can determine how to model new issues raised by the previous results and then proceed back to the exploration phase to further refine the data.

Using the sEMMA-based mining method, once you have developed the champion model, it needs to be applied to score new customer cases. Product implementation is the end result of data mining — the final step of realizing the return on investment from the mining process. Enterprise Miner automates the phase of deployment by providing scoring code in SAS, C, Java, and PMML. Not only does it capture the code for analytic models, but it also captures the code for pre-processing activities as well. You can score your production data seamlessly on another machine and deploy the scoring code in batch or real-time on the Web or directly in relational databases. This results in faster implementation and allows you to spend more time evaluating and developing existing models.

sEMMA- A Big Data Era

How does sEMMA methodology affect the Big Data era? The short response is that it is not largely so. SEMMA is a logical process that can be followed, whatever the size or complexity of the data. But in sEMMA, the 's,' or sample, is less likely to be as critical with the more powerful data mining systems available. From my experience working with big data, sEMMA can be used to address very large analytical databases.

Classification of Binary

Binary classification is, from my experience, the most common kind of predictive model. Key corporate decision-makers and other organizations often have to make critical decisions quickly, requiring a system to arrive at a confident yes / no decision. Such systems are not designed to do things in a partial way or with a certain probability. Either they do or they don't. This is illustrated very well in the famous checklist "go or no go for launch" that the controller of each system has to answer for. A launch can only proceed after each device has been tested by the flight director and has obtained the response "go for launch." Many decisions in the business world are binary, such as should I extend credit for this car purchase, or will you respond to this marketing campaign.

The probability that an event will or won't occur is also very useful information with predictive modeling of a binary target. To illustrate this point, let me provide an example. We establish a black-and-white situation with a binary target; will it rain tomorrow — yes or no? Either there will be rain or it won't, but you're very unlikely to see your local meteorologist (or even the National Weather Service) give a 0 percent or 100 percent chance of rain because while it will either rain (100 percent) or not rain (0 percent), the estimate is implicitly the degree of confidence we have in our prediction. This calculation of confidence is always much more accurate than the estimation of binaries themselves. Does your behavior differ when you see a 10 percent rain chance versus a 40 percent rain chance? Are you able to put your convertible windows down or the roof off? All of these forecasts mean that it is less likely to rain than not, as the percentage is less than 50 percent. However, from experience with having wet car seats for my home commute from a day when it was unlikely to rain (30 percent), I keep my windows up if there is any chance of rain because the downside risk is far greater than the upside potential.

A frequently seen case in binary predictive modeling is when you are dealing with a "rare event." What qualifies as a rare event varies depending on the industry and the domain, but it universally means that the event occurs with extremely low probability and may require some special treatment in the analysis.

Let us explore the unusual occurrence more closely using an example from golf sport.

1. The most sought-after occurrence in golf is the hole in one. This is where, during at least a 9-hole round, a golfer takes a tee shot and makes it into the hole in one shot. When you make a hole in one at certain competitions, you will always win a car, and tradition dictates that you buy a round of drinks at the clubhouse for everyone irrespective of when it happens.

2. To be legal, someone else must witness the shot. The U.S. Golf Association (USGA) maintains a record of all holes in one and estimates the odds of scoring a hole in one as 1 in 33,000. So, while the odds of anybody hitting a hole in one are very small (.003 percent), some people, like Tiger Woods, have not only hit one but hit multiple holes in one. Tiger has 18 holes recorded in one but is not leading in that category. The most recorded holes in one go to Norman Manley, with 59. To put mathematical terms around the chances of hitting a hole in one, we'd say no one would possibly hit a hole in one. Assuming 1 in 33,000 tee shots results in a hole in one and (after removing long holes) that there are 14 holes, you could reasonably make a hole in one, if you golfed around every day of the year, you'd hit a hole every seven years on average.

Once we have determined the probability of the occurrence of the event for each observation, they are then sorted in descending order from the highest probability to the lowest probability based on the probability of the experience of the event. Using the above example, if we were marketing commemorative hole-in-one plaques, we would apply the model to all golfers in our database, sorting the list from the highest likelihood to the lowest likelihood of making a hole in one. We would then send an offer letter from the top of the list to all the golfers until our marketing campaign budget is exhausted or to those who have exceeded a certain threshold. It will ensure we give the deal to the golfers, based on the layout, for a commemorative plaque, most likely to find a hole in one and who will be interested in our services.

Classification Multilevel

Multilevel classification or nominal classification is very similar to binary classification except that there are more than two stages today. Nominal marking is an extension of binary classification. There are some examples where nominal classification is common, but this is the rarest of goals for the most part. An example of such a pattern can be seen when looking at consumer turnover in the mobile industry. A firm may not only be interested in the binary response of whether or not an account remains active. Rather, it may want to dig deeper to look at a voluntary churn negligible response (customer wants to cancel the contract), involuntary churn (e.g., a contract was terminated because of overdue payments), or an active customer.

In many cases, when an exceptional case is added to what could be a binary decision, a nominal classification problem arises. This happens, e.g., in the case of avoiding credit card fraud. When a credit card transaction is started, there is a very short window for the card issuer to accept or decline the transaction. Although this can be viewed as a simple binary problem, accept or reject, there are some transactions where the decision is not so straightforward and may fall into a gray area. The third level of response can be added to suggest that further analysis of this transaction is required before a decision to approve or decline can be made.

The problem of nominal classification presents some potential problems from both a statistical and reporting perspective. Because of simply estimating one event probability (P) and then taking 1 – P to arrive at the likelihood of non-event, you need to measure the likelihood of event # 1 (P1), the probability of event # 2 (P2), and so on until

The last stage to be measured using $1 - - \text{range} = -11$ P. Computing the rate of misclassification often poses a challenge. Since there are many choices, the values of the report must be calibrated so that the reader can easily interpret them.

Interval Prediction

The final method of prediction is the prediction of intervals, which is used when the goal amount on the number line is continuous. Salary is an example of a forecast that employers and workers alike would like to precise. The casualty insurance industry is a field with a number of predictive interval models. When you buy a vehicle in the U.S., you need insurance for it. You've already received a quote from several different insurance agencies to get the policy, and each one has given you a slightly different price. This price discrepancy is due to the various predictive models and business factors used by each insurance firm. The business considerations are the amount of exposure in other regional markets or an overall mandate that aims to gain market share in either regional or economic strata of a given sector.

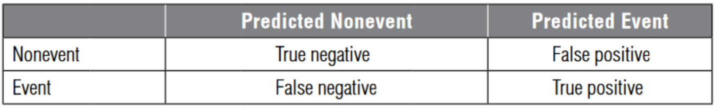
Insurance firms typically use three different types of predictive interval models, including claim duration, severity, and pure premium. For each of these models, the insurance company will make predictions based on their historical data and your specific information, including the car's make and model, your annual driving mileage, your past driving infringement history, etc.

Assessment of The Predictive Models

In the process of developing predictive models, one factor would be deciding which model is best. In addition to the chosen algorithm and its related parameters, a model consists of all the transformations, imputations, variable selection, variable binning, and so on manipulations that are applied to the data. For any realistic data mining problem, the vast number of choices and combinations renders a brute-force "try all" approach impossible. Therefore, the topic of model validation plays an important role in selecting the right one. Model evaluation, simply put, is trying to find the best model for the provided data for your application. The complexity comes from the term "best." Just like buying a washing machine, there are a number of aspects to consider, and not everyone agrees on a single definition of "best." The common set of measurements for model evaluation are listed and defined next. Consider, for example, a local charity that organizes a meal drive. It has a donor mailing list of 125,000 people and past campaign data, which will be used to train the model. We split the data so that there are 100,000 individuals in the training section, and 25,000 in the validation section. Both partitions have a 10 percent response rate. The validation section, which is a best practice, will be used for evaluation.

Categorization

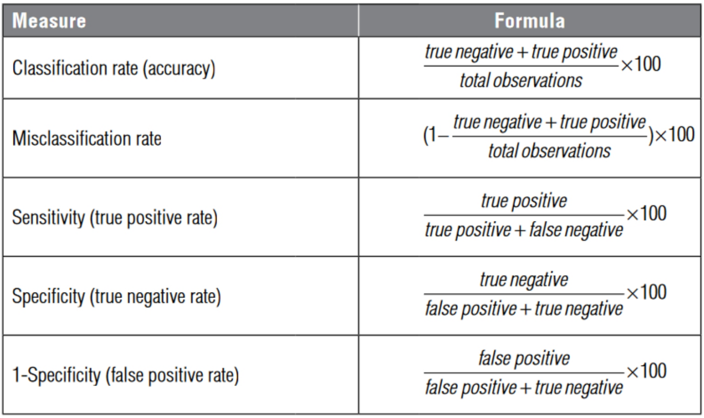
There is a set of measurement criteria based on the matrix of 2x2 decision, as shown in Table 5.1.



	Predicted Nonevent	Predicted Event
Nonevent	True negative	False positive
Event	False negative	True positive

Table 5.1 Decision Matrix for Model Assessment

Classification is a common approach because it's simple to understand, it closely aligns with what most people equate with the "best" model, and it tests the model fit across all values. If the proportion of events and non-events is not roughly equal, then the values have to be adjusted to make the right decisions. (See Figure 5.2.)



Measure	Formula
Classification rate (accuracy)	$\frac{true\ negative + true\ positive}{total\ observations} \times 100$
Misclassification rate	$(1 - \frac{true\ negative + true\ positive}{total\ observations}) \times 100$
Sensitivity (true positive rate)	$\frac{true\ positive}{true\ positive + false\ negative} \times 100$
Specificity (true negative rate)	$\frac{true\ negative}{false\ positive + true\ negative} \times 100$
1-Specificity (false positive rate)	$\frac{false\ positive}{false\ positive + true\ negative} \times 100$

Table 5.2 Formulas to Calculate Classification Measures

Operating Receiver Characteristics

The operating characteristics of the receiver (ROC) are determined for all purposes and are shown at analysis graphically. The ROC plot axis is the Sensitivity and 1- Specificity, determined from the levels of classification.

LIFT

Lift is the ratio of right responders to baseline response levels. To measure lift, a percentile must be added to it in the results. It is generally referred to as file size, and the first or second decile is usually picked. For the example of the food drive, if we measure the lift at the first decile (10% of the data), the baseline (or random) model will have 2,500 respondents to the campaign so that 250 respondents (2,500 u.1) would be in the first decile. Our model is good; it captures 300 respondents in the first decile so that the lift at the first decile is 1.2 (300/2500 = 12 percent of the response captured/10 percent of the response). I like to use the cumulative lift for my model evaluation as it is monotonous, and in practice, campaigns should sort a list with the possibility of responding and then sell until a natural break is detected or the campaign budget is depleted.

The Criterion of Knowledge for Akaike

The Akaike information criterion (AIC) is a statistical measure of fitness goodness for a specific model. It maximizes the $-2(LL + k)$ expression, where

K = number of approximate parameters (The number of terms in the model for linear regression)

LL = maximized value of the log-likelihood function for the model in question.

The smaller the AIC, the better the fits the data model. Since

The term k, the smaller number of parameters of the model, is preferred. AIC values can be negative, but in a real-world data mining problem, I don't remember ever seeing it.

5.1 Common Predictive Modelling Techniques

The main part of data mining and, in my view, the most fun is after the data has been processed, and the data scientist is able to run a "test tournament."

Modern data mining problems the best approach is to carry out a variety of different computational techniques or algorithms and attempts to use various settings or parameters within a single algorithm. The explanation for so many trials is that there's an element of brute force that needs to be get more practiced to gain the best result. Depending on the specific domain and the modeling challenge, the number of iterations required to achieve the best model will differ widely.

It is essential to note that although this chapter focuses on predictive modeling or supervised learning, both of these important strategies can be used to classify the hidden structure of a collection of data in unsupervised approaches. Supervised learning has a target variable, and as many of the results as possible are identified or estimated correctly by the methods. Supervised learning has consistent indicators and qualitative evaluation of the model. Uncontrolled learning, on the other hand, lacks a goal and thus good objectivity in evaluating the consistency of the model.

Each of the following parts covers a specific modeling technique and has the overall progression of a brief history on the methodology, a basic illustration or tale to explain how the process can be used followed by a high-level mathematical approach to the process and finally a section of references for you to obtain more information.

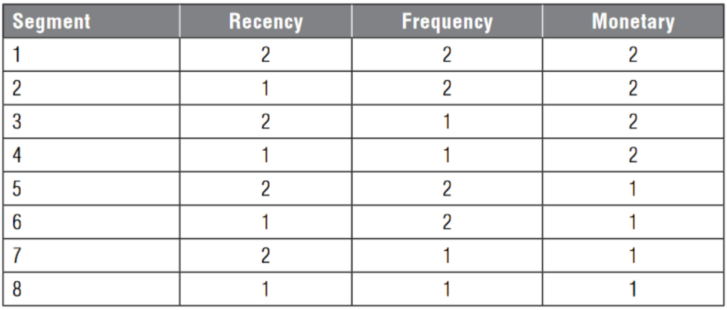
Lastly, many of the most common predictive modeling techniques are discussed in this chapter with examples, graphics, and some mathematical information. It gives you a description of the methods and their applications.

RFM

RFM, or recency, frequency, and monetary modeling, is a very basic but efficient way of rating clients. It explores how long they have been investing (monetary) since a custom-s last visit (recency), how much they buy (frequency). This approach is more attractive because it does not require advanced software or "fancy" algorithm expertise. The approach can be clarified easily to all levels of information about management and the domain—the tool most widely used in the marketing of databases or direct marketing. One basic approach is to take each metric and break it into parts. These may be data-based, such as the deciles of how much households spend, or they may be regulated to comply with standard accounting practices, such as the recency of 0–90 days, 91–180 days, and 181 + days. If we take only two rates in each dimension as a basic example, then we will have eight segments, as shown in

Table 5.1. Typically, the number of segments can be determined using:

Table 5.1 RFM Segmentation with Two Groups per Variable



Segment	Recency	Frequency	Monetary
1	2	2	2
2	1	2	2
3	2	1	2
4	1	1	2
5	2	2	1
6	1	2	1
7	2	1	1
8	1	1	1

All clients that fall into one of these categories. Through segmenting the clients, a particular program or procedure can be tailored to better optimize pt. Looking at the divisions from Table 5.1, it is clear that the most important group of customers is Segment 1. They've made a purchase recently, they buy regularly, and they're spending an above-average amount relative to the typical consumer.

Using Table 5.1 segment assignments, it is clear that segment 1 is the most valuable, and segment 8 is the least valuable. Nonetheless, there is little clarification in the other categories, so profiling the categories is normal in practice, so likely merging some if there are not enough differences.

The typical segmentation of the RFM is to use five bins, one for each quintile, for a total of 125 segments per component. It is advisable not to use the RFM prediction method because although it is reliable in certain situations, it is merely a historical indicator. It does not have the ability to compensate for changes in macroeconomic conditions or transformations in human life, such as birth, death, divorce, or retirement, which would dramatically change the marketing plan.

A few differences in this section between RFM and the other techniques are that RFM doesn't generate any form of score code that can be applied to new observations. It is in contrast to other approaches, such as decision trees, where a major contribution is to deploy the main model into a production system that can score incoming records very quickly or may be used to score a wide table of customers. In the case of RFM, the whole cycle has to be replicated when adding new data to the data collection. The difference is that scores cannot be compared across data sets due to the lack of score code and method repeat. If you look at the 555 categories (which represents the best of the best customers in a traditional setting) from a December sales consumer retail situation, that'll likely be a very different set of customers with different profiles and credentials than the May sales customers.

The number of sales, their numbers, and their duration for the Christmas holiday season vs. May are somewhat different. There are two key ways to use the RFM ratings, and they vary based on how you believe that recency is more important than duration, which is more important than monetary. If this is the case, then a simple data sorting would allow the most valuable customers to be targeted, and you can go down the list sequentially until your campaign budget is run out. The reason for believing that recency is most important is that the longer it takes for your clients to return to their company, the less likely they will ever return. It's easier to influence the actions of the customer how much they come or how much they spend, but a long time after the last interaction raises the risk that you've already lost the high-value client. Conversely, if you do not believe that the restriction may apply for a variety of reasons, then a score of 345, 534 or 453 is all approximately equal, and you can add up the individual score digits (12) in this case and sort the list by the summary score starting with the customers getting a 15 and continuing the list down until the campaign budget is depleted.

A variety of variations on this approach may also be added to add some degree of complexity, such as secondary sorts or multiplying the individual scores (R, F, or M) by a weighting factor to represent each relative value. A final aspect required for RFM and many of the modeling techniques is the preparation of data before the technique can be implemented. The sales database is full of transactions in a retail setting, which have relatively few details: usually only date, time, account number, store ID, and amount. Usually, there will be a transaction ID linking to a specific table in the enterprise data warehouse, which includes a complete shopping cart list. But all of this data is transactional, and RFM requires customer-level data, so the process of collecting, transforming, and www.it-ebooks.info load (ETL) (discussed earlier in Part One) to build the customer-level data set is summarized. This approach is based primarily on the Pareto principle, which is generally described as rule 80/20. This explains that just 20 percent of your customers make up 80 percent of your sales. The name comes from an observation made in 1906 by the Italian economist Vilfredo Pareto that only 20 percent of the population owned 80 percent of Italy's land. This has contributed to the distribution of Pareto, which is a transfer of power, and inter alia has applications in many measurable phenomena in the social and actuarial fields.

REGRESSION

Regression analysis is probably the first predictive modeling approach you studied during your undergraduate studies as a practitioner or the most popular modeling tool for your analytics community. The conceptions of regression were first presented by Adrien- Marie Legrendre and Carl Gauss in the early 1800s. Born into a wealthy French family, Legrendre has contributed to a variety of developments in mathematics and statistics. Gauss, by comparison, had been born in Germany to a poor father. Gauss was a child maths prodigy, but during his life, he was hesitant to publish any work that he felt was not beyond scrutiny. Several of his personal journals were discovered after his death, which described his ideas and thoughts. Such theories were important developments in mathematics. If Gauss had been more successful in publishing his thesis, it is estimated that he could have progressed the field of mathematics by more than 50 years. The two smart minds of the early twentieth century, along with Sir R.A. Fischer, describe the regression process. Francis Galton (Charles Darwin's Cousin) coined the term "regression" to describe the evolutionary phenomenon of extreme values shifting toward a mean population. A typical example of this is children of tall (above average height) parents "regressing" towards population mean and becoming shorter than their parents; vice versa, children of short (below average height) parents are taller than their parents. Before Galton, the methodology was called the "least squares." In my perspective, regression is today's most powerful factor in making business decisions. The regression analysis has many valuable features; one is that tests are easy to interpret. Regression principles are well understood, and the technique is well-developed in such a way that a professional practitioner's well-tuned regression model will outperform other algorithms that gain attention from the discipline of machine learning.

5.2 Regression Model Assumptions

It's necessary to remember the assumptions of a linear regression model. It is also necessary to note that some, if not all, of the premises, are broken to some degree in data mining practice. There are ways to resolve these assumptions by using more advanced techniques of regression modeling, discussed later in this section. The typical linear regression suppositions are:

There is no measuring error in the target's measured values and each input variable. Essentially the values in the data set are assumed to be accurate.

The relationship between the variable dependent and all the independent variables is linear.

- The comments are separate.
- Independent variables are not linear combinations.
- The variance of the most dependent variable is constant (homoscedasticity)

An additional assumption exists for logistic regression:

- The actual conditional probabilities are the mathematical feature of the independent variables.

Additional Regression Techniques

Here is a list of advanced regression techniques that are useful for large-scale data mining. Observations would be more serious and relationships more complex between variables.

- Adaptive regression suits multivariate models of adaptive spline regression. This is a technique of nonparametric regression, which incorporates both splines of regression and methods of model selection.

The technique of regression will employ both continuous and discrete inputs.

- Generalized additive models (GAMs) are nonparametric, meaning that linear predictors are not needed. GAMs make the regression variables smooth functions. This technique is particularly useful for fitting non-normal data with additive models. The technique can use continuously as well as discrete inputs.

- Locally approximate scatter plot smoothing (LOESS) is used to suit nonparametric models using the local regression method. This approach is good for modeling data from an unknown distribution, and it is probable that and outliers are present in the data.

- Orthogonal regression uses the computational method Gentleman- Givens to perform regression on unconditioned results. Also, the Gentleman- Givens approach offers more reliable estimates of parameters than non-specialized methods of regression.

- Aside from having a minimal prediction error for a response variable, a partial minimum square regression often balances this with the variance of the predictor or input variables. Sometimes this balance is accomplished by using linear combinations of variables such as those used in components of the theory.

- Proportional hazard regression uses the Cox models to match the data for survival regression.

- Quantile regression is used for modeling covariate effects on conditional-response variable quantiles.

- Robust regression uses the estimation of Huber M and high breakdown value to suit models.

This method of regression is good for identifying outliers and making reliable predictions when there are outliers in the results. Applications in the study of the Big Data Age Regression is a basic predictive modeling approach for a statistics student, but it is potentially a new and revolutionary concept for those coming from other disciplines, such as computer science. I was just watching a talk in preparation for a presentation on machine learning. The speaker, a well-educated professional from a prestigious university, said: "For those of you who are not familiar with this, we are using a technique called logistic regression. ."It was a reminder to me that there is a very wide field of data mining, and people approach problems from various viewpoints. Through my experience working with customers, even in the Big Data era, regression is still the predictive modeling approach that dominates. Which I consider can be related to a number of factors. The first is that performance is simple to understand. The parameter estimates are very open to people with a wide variety of quantitative expertise and market experience resulting from a regression analysis. Second, tests may be used to assess the model's key factors (sometimes referred to as driver variables or key performance indicators [KPI]). Third, the methods are (or should be) parallel; they can be used on almost any size problem. Regression is still a very actively researched area, and modern lasso techniques, robust regression, and ridge regression have all advanced the field based on concepts developed more than 200 years ago. I believe that this area will continue to push forward and produce new developments as researchers and practitioners alike are searching for new forms of regression to provide insight into the data relationships. In this era, where data volumes continue to increase, and data are combined in more ways, and at lower granularity rates, the need for advanced regression techniques will remain in demand for a long time to come. Any package of tools that you are considering for managing big data will do regression. If not, then that's a big red flag.

Generalized Linear Models (GLMs)

In 1972, statisticians John Nelder and Robert Wedderburn, who worked together at the Rothamsted Experimental Station in England, described generalized linear models (GLMs) as a unified system for probit4models in the pursuit of chemical dosage tolerance, contingency tables, and much more. This generalized definition of models removed the limitations, which were prohibitive for some types of problems, and provided flexibility to accommodate non-normally distributed response variables, provide a small range mean and/or non-constant variance — all of which are violations of the OLS regression assumptions. GLMs are composed of three components:

1. Any distribution from the exponential family can be the probability distribution of the response variable Yi (the random component).

2. The linear model that includes the explanatory variables and the parameters of the model, xi′β(systematic component).

3. The connection feature that defines the relationship between the components systemic and random.

4. More precisely, the normal linear portion of a linear model, xi′β, now links to the expected valueμiof the response variableYi by gxii()μβ=′ function. The relation function can be any differentiable, monotonic function.

5. For ordinary linear regression, the relation is just the identity, and the distribution of the answer is normal. For the Bernoulli-distributed response variable, a logistic regression model, another example of a GLM, uses the logit relation function; in addition, this provides a probability estimate between 0 and 1 for the model outcome.

6. This model should be used when the model tries to forecast turnover, attrition, or some other binary event in which the interesting event is not as relevant as the probability of the event occurring.

Using a Poisson or negative binomial distribution and a log link function, counts can be modeled in a GLM. Types of count models may include the number of claims that someone makes against an insurance policy, the number of vehicles that pass through a toll booth, the number of items that will be returned to a store after purchase, or some other counting-type method in which the frequency of one event is independent on another.

The Big Data Applications

Throughout the insurance sector, a key use of GLMs as regards big data is for building risk generating models. Rate-making is the determination of insurance rates based on risk characteristics (captured in rating variables); GLMs have become the norm for fitting rate-making models because of their ability to simultaneously accommodate all rating variables, eliminate noise or random effects, provide diagnostics for fit models and allow interactions between rating variables. Usually, the response variables designed for rate generating applications are claims counts (the number of claims a person files) with a Poisson distribution, claim amounts with a gamma distribution, or pure premium models using the Tweedie distribution. These distributions are all lead to members of the exponential family, and the log connection function usually relates the response to a linear combination of rating variables, which allows the formulation of these models as GLMs. One major US-based insurance firm uses GLMs to create data-based rate-making models that include 150 million policies and 70 degrees of freedom.

5.3 Neural Networks

In the almost last half of the twentieth century saw the development of artificial neural networks. Researchers were inspired by a basic mathematical model of a biological neuron — the central structure inside the brain that allows for animal learning. In this model, defined in 1943 by neurophysiologist Warren McCulloch and logician Walter Pitts, a neuron "fires" (i.e., it’s out-set becomes 1) when the sum of its exciting inputs reaches a threshold value and no inhibitory inputs are 1. McCulloch and Pitts showed that the networks of these model neurons could measure the value of most complex logical equations (equations where only the values 0 or 1 are present in the variables). Their initial definition is represented in Figure 5.11.

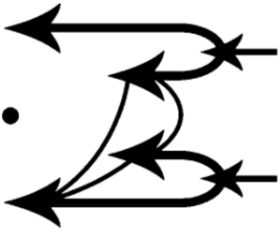


Figure 5.11 Original Drawing of a Neural Network
Source: McCulloch, W. S. and Pitts, W. H. (1943). "A logical calculus of the ideas immanent in nervous activity." *Bulletin of Mathematical Biophysics*, 5, 115-133.

In the late 1950s, Frank Rosenblatt used a network of neurons similar to McCulloch- Pitts neurons, which he called a perceptron, to correctly classify digitized images of letters by measuring the inputs to the neurons and then modifying those weights using the Perceptron learning algorithm. The Perceptron learning algorithm begins with a random collection of weights and then iteratively modifies those weights based on an error function of the difference between the actual output of the neuron and the desired output of the neuron for each neuron in the network (1 if this neuron represents the digitized letter and 0 otherwise). Another type of training method is also called supervised learning since it sets the network's weights by looking at a set of examples where the network's target output is defined. Once subsequently faced with a set of inputs where the correct output is not known, it is assumed that the "qualified" network would generate the correct output. (See Figure 5.12.)

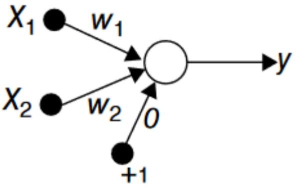


Figure 5.12 Simplified Perception Diagram

Sadly, a perceptron can only be trained to distinguish patterns belonging to linearly separable classes — classes where all patterns can be properly separated by a hyperplane in the input space, as shown in Figure 5.13.

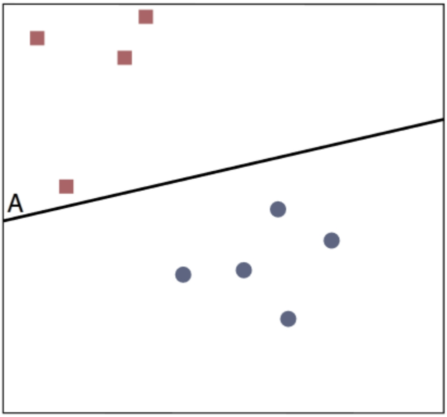


Figure 5.13 Data Properly Separated by a Hyperplane

In 1969, Marvin Minsky and Seymour Papert pointed out the lack of capacity to model nonlinear interactions in data in the book Perceptrons. The book was controversial when its study of perceptron limitations was perceived to mean that even extended neural networks would be stopped from solving more complex, real-world issues. This proved to be incorrect, but this interpretation significantly delayed research into neural networks for over a decade as research into artificial intelligence shifted away from connectionist algorithms (representing information as weights and connections) to symbolic algorithms (representing information as logical proposals and using logical inference rules to deduce results). In his summary of the third edition of Perceptrons, "No Harm Intended," from 1989, Jordan B. Pollack provides an insightful study of the debate over perceptrons.

Throughout the latter half of the 1980s, the breakthrough for neural networks came after differentiable squashing activation functions (e.g., logistic, etc.) replaced the nondifferentiable phase activation function used in perceptrons. In addition to allowing neural networks to model arbitrarily complex functions, the differentiable activation functions allowed the use of the backpropagation method for calculating the weight gradient of the error function, which in turn permitted the use of the gradient descent algorithm to determine the optimum weights. This approach was defined by David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams in a 1986 paper entitled "Learning Representations by Back-Propagating Errors."

Eventually, in answer to the Perceptrons book in 1989, Kurt Hornik, Maxwell Stinchcombe, and Halbert White's paper "Multilayer Feedforward Networks are Universal Approximators" showed that a three-layer (input, hidden, output) neural network, where the hidden layer consisted of neurons with a squashing activation function, could "approximate any function of interest to any desired degree of activation."

As we sit here in early 2014, neural networks are being used in a wide range of fields to model everything from credit card fraud, where best to drill an oil well, what stocks to invest in, how to find faces in photos, whether a person is likely to respond to an ad, whether a person is likely to catch a disease, and whether a drug is likely to cure it.

Notwithstanding some of the excitement surrounding neural networks, where they are represented as human learning, they simply reflect a straightforward problem of nonlinear regression.

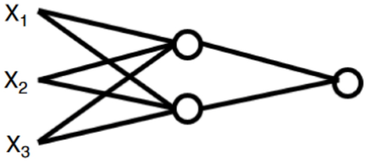


Figure 5.14 Standard Three Layer Neural Network

A nonlinear function of the in-placed variables works in a nonlinear regression problem by optimizing the weights of a nonlinear function y(W,X) where
W = vector with weight to be determined
X = known input vector variables
Y = Known Output

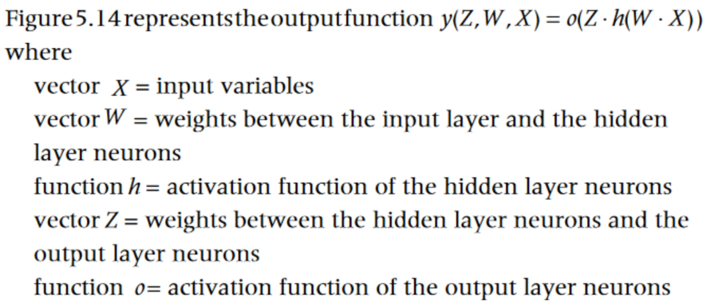


Figure 5.14 represents the output function $y(Z,W,X) = o(Z \cdot h(W \cdot X))$ where

vector X = input variables
vector W = weights between the input layer and the hidden layer neurons
function h = activation function of the hidden layer neurons
vector Z = weights between the hidden layer neurons and the output layer neurons
function o = activation function of the output layer neurons

A neural network is simply a particular type of nonlinear system, and what is commonly known as a graphical representation is simply a diagram of that system.

Three layers (input, hidden, and output) are the most commonly used neural networks, feed-forward networks use squashing activation functions for the hidden layer neurons. Neural networks with five or more secret layers have recently been found to be useful in the identification with features in images and in the processing of natural language.

These networks, extended to include several hidden layers, also known as Multilayer Perceptrons (MLPs). Nevertheless, the term "neural network" has been extended to a number of other architectures, such as auto-encoders, coevolutionary neural networks, radial base function networks (RBF), and self-organizing maps (SOM).

Usually, multilayer perceptrons are used for two forms of problems, which generally specify the number of neurons placed in the output layer, the activation function used by the output layer, and the error function used to accumulate the discrepancies between the output and the desired output.

Problems of regression are those where you seek to predict a numerical value in response to the given input vector (like predicting a person's age based on weight, height, and sex). Since you are attempting to predict a single number, the output layer contains only one neuron. The activation of identity output function is sometimes used for the output layer neuron, and the difference between the network output and the desired output is generally used for the function error.

Classification problems are those in which you seek to determine if a given input vector belongs to one of many groups (such as predicting a plant species given some of its measurements). In comparison to regression problems, where the output layer has a single neuron, the output layer typically has one neuron per possible class. In conjunction with the cross-entropy error function, the softmax activation function is also used for the output layer neurons, since the output value can then be interpreted as the probability that a given input vector belongs to the class defined by each output neuron.

Figure 5.15 represents the two output functions
$y_1(Z_1, W, X) = o(Z_1 \cdot h(W \cdot X))$, and $y_2(Z_2, W, X) = o(Z_2 \cdot h(W \cdot X))$

where
vector X = input variables
vector W = weights between the input layer and the hidden layer neurons
function h = activation function of the hidden layer neurons

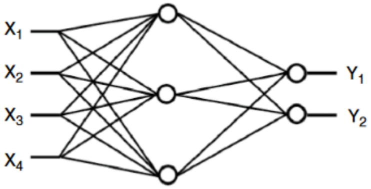


Figure 5.15 Two Output Functions

-- output neuron, in this case, may represent one of two possible classes to which each input vector can belong. The output may be interpreted as if the output y1 was greater than y2; an input vector X belongs to class1.

In order to prepare a neural network, you need a framework for calculating.
Weights that minimize the extremely nonlinear objective function for a collection of exemplary preparation. Note, the objective function is not the output of the network — it reflects the network error — the absolute discrepancy between what the network should be outputting and what it is outputting for all outstanding preparation.

Normally the task of determining the minimum value of a nonlinear differentiable function is achieved by using either the gradient descent method or Newton's method of some kind. All of these approaches allow you to recalculate the objective function gradient repeatedly. In the sense of neural networks, the gradient in relation to increasing weight is the vector of partial derivatives of the objective function. Within this vector, the value of each partial derivative reflects the sum that the objective function adjusts with a small weight change. A straightforward method for measuring this gradient efficiently was the main contribution of the backpropagation algorithm.

The gradient descent approach uses the notion that if you start at randomly selected initial points (a randomly selected initial collection of weights) and travel gradually in the direction of the negative gradient of the objective function (in the steepest downward direction), you will inevitably end up at the edge, or at least at a position where the gradient is zero (a valley). The reason you need to push slowly is that you don't have to skip over the absolute minimum and backtrack.

The reason the function needs to be differentiable is, so the gradient exists, as it does with standard activation functions for multilayer perceptrons, but not, for example, the activation function for phase. You have to determine how far along the negative gradient to travel at each step-the step length-while using the gradient descent process. One the other way to do this is to transfer a proportional sum to the gradient. Below is the gradient descent formula,

$$W_{n+1} = W_n - \lambda \nabla f(W_n)$$

where

$W_n =$ current vector of weights

$\lambda =$ learning rate

$\nabla f =$ objective function to be minimized

Although heading towards the negative gradient gives the steepest descent from the current location, it is not necessarily the path that gets the fastest to the ultimate minimum.

If used for minimization, Newton's method uses the first derivative, the gradient, and the second derivative, the curvature, to find the minimum value of a function, and is considered a method of second order. It calculates a simple quadratic function at each step with the same gradient and curvature as the current objective function and then seeks new weights, which would reduce the quadratic function. Even these new weights can contribute to a decreased objective function. This then requires the cycle until the objective function gradient is near zero. Below is the Newton Theory Formula:

$$W_{n+1} = W_n - H^{-1} \nabla f(W_n)$$

where

$$H = \text{Hessian}^{11}$$

Second-order methods typically converge to a minimum with far fewer iterations than first-order methods such as gradient descent, but the reverse-hessian calculation is time-consuming, on the order of n3 operations for n variables (weights) function. To ameliorate this problem, methods were developed, called Quasi-Newton methods, which iteratively estimates the inverse Hessian. Possibly the most basic and common of these is the BFGS (Broyden – Fletcher – Goldfarb – Shanno) process. This approach, like the nonlinear conjugate gradient method, first calculates a search path (by multiplying the approximate inverse Hessian by the current gradient), and then uses an incorrect line search along that path to find a new set of weights that result in a reduced gradient. Nevertheless, the store the Hessian still needs n2 storage locations. The Limited Memory BFGS (L-BFGS) method was developed to overcome this last problem.

Some important considerations when dealing with neural networks are:

Starting Values: Where a neural network's weights are close to zero, a linear regression model approximates the model. Over several iterations, the neural network model becomes more and more nonlinear as weights shift and help to describe nonlinear relationships that are not observed. When the starting values are too close to zero the algorithm does not converge to the true values. Likewise, beginning with broad weights usually results in weak solutions. The challenge is to select starting values that are similar to zero but are not trapped there.

Overfitting: One of the greatest worries is overfit- ting when using historical data to create a model that forecasts future results. Overfitting in this context means that the model has been trained to match very well a given set of training observations, but does not generalize well — does not do well to predict the effects of observers not being used for training. Unlike linear or logistic regression, where you use historical data to measure only a few model parameters, with neural networks depending on the number of input variables and the number of hidden neurons, hundreds or even thousands of model parameters (weights) may be used to match the previous data. Since neural networks are universal approximators, they should be able to correctly predict the outcomes of the training observations by adding enough hidden neurons and running the training algorithm for enough iterations. But what the neural network learns during training is the noise in the particular data set of training as well as the underlying process that generated the data. When new data are introduced, there will be no the same noise.

Validation data for the overfitting modification: One strong answer during preparation is to use cross-validation. You split the training data into a validation subset and a testing subset. At least 25 percent of the available data should be included in the validation section. You must ensure all subsets have the same data discharge. You only send the training subset to the training algorithm. You measure the prediction error on both the training and validation subsets after each iteration of the training algorithm, using the current collection of weights computed by the training algorithm. What you will usually see is that both the training subset error (training error) and the validation subset error (validation error) both begin down together as the training algorithm performs its work. So, validation error starts to go up again at some point, even though the training error continues to go down. At the point, you've achieved the best-generalized model, the validation error increases.

Scaling Input data: The input variables scale influences the weights in the initials. Once you start constructing your model, it is best practice to scale the inputs to a mean of value 0 and a standard deviation of 1. It would ensure fair and equitable treatment of all input variables. Scaling the input variables also helps in finding appropriate starting points.

Global Minimum Assurance: If the aim is to minimize the error is nonconvex (meaning that in hyperspace, every line between two points is not necessarily a valid solution; think of Swiss cheese, where air bubbles are invalid solutions), it is possible to get stuck at a minimum local level and never hit the target overall. The ability to identify the global minimum is primarily dependent on the starting values.

Core Example of Neural Networks

Sir Ronald Fisher obtained data about three forms of Iris. He gathered their petal length, petal width, sepal length, and sepal width measurements. Using these four interval scales, the goal is to correctly identify the three Iris virginica Versicolor and setosa species from these measurements.

The first step is to segment the data to ensure the neural network model is not Overfitting, so we take a stratified sample and divide the data 70 percent for training and 30 percent for validation. The network configuration is composed of four input neurons, five secret neurons, and three target neurons, as shown in Figure 5.16.

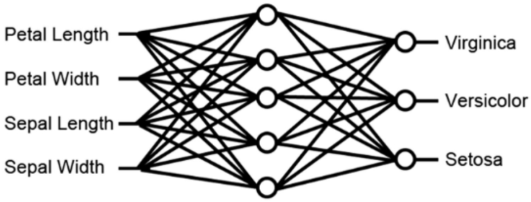
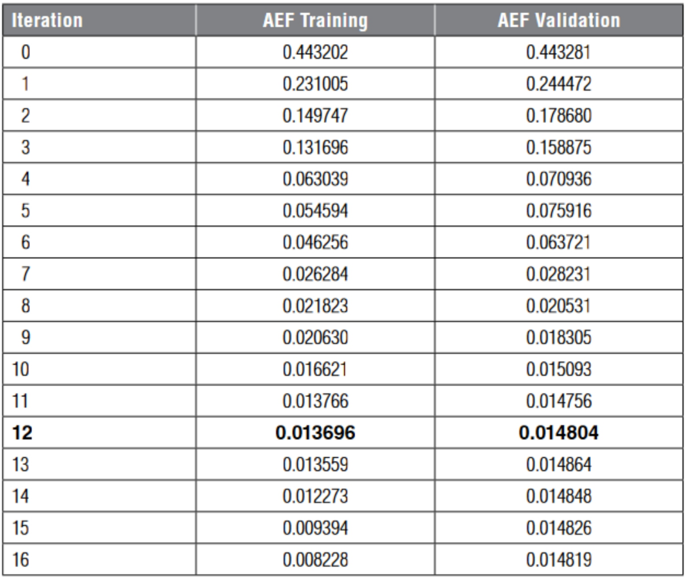


Figure 5.16 Neural Network Diagram for *Iris* Data

The initial weights are initialized at random between the input neurons and the hidden neurons. The Objective function error term is shown for 16 iterations in Table 5.2.

Table 5.2 Average Error Function (AEF) of Training and Validation for Neural Network for *Iris* Data



Iteration	AEF Training	AEF Validation
0	0.443202	0.443281
1	0.231005	0.244472
2	0.149747	0.178680
3	0.131696	0.158875
4	0.063039	0.070936
5	0.054594	0.075916
6	0.046256	0.063721
7	0.026284	0.028231
8	0.021823	0.020531
9	0.020630	0.018305
10	0.016621	0.015093
11	0.013766	0.014756
12	**0.013696**	**0.014804**
13	0.013559	0.014864
14	0.012273	0.014848
15	0.009394	0.014826
16	0.008228	0.014819

From the mentioned table, you can see that the training error keeps decreasing as the weights are adjusted and balanced. Nonetheless, the validation error begins to grow after the eleventh iteration (as seen in Table 5.2 as bold). It is because the neural network has now overfitted the model to the training data and has lost its predictive ability overall. The model to be used in classifying the Iris is the one generated at the 11th iteration.

The neural network was able to accurately and correctly identify all but one finding in the validation data as seen in the third row of Table 5.3 in which a virginica iris was erroneously labeled as a versicolor iris. This error refers to a validation data misclassification rate of 2.2 percent.

The word "deep learning" has recently become widely used to describe machine learning approaches that can learn more abstract concepts than previously available methods, especially in the fields of image recognition and text processing. For example, in tasks such as recognizing handwritten digits (see Figure 5.17), or selecting images with dogs in them, neural networks with three or more hidden layers have proved to be doing very well.

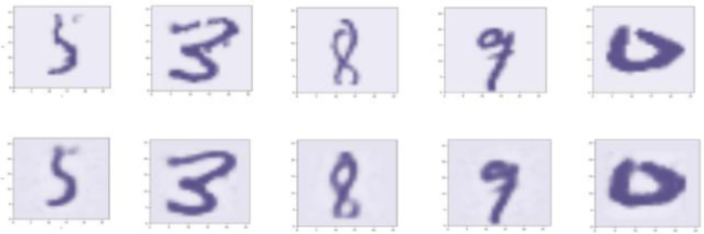


Figure 5.17 Improvement of Digit Recognition Using Deep Learning

Training these neural network forms has proved quite a challenge. The standard backpropagation algorithm begins to fail on networks with large numbers of input variables (each pixel of an image is usually a single input variable) and several hidden layers because of the gradient of the total error in the initially hidden layers is extremely low and noisy with respect to the weights. This is known as the problem of vanishing gradients. Furthermore, a large number of input variables, resulting in very large numbers of weights that have to be determined, make the typical optimization process very sluggish.

Geoffrey Hinton and R., in 2006. R. Salakhutdinov explained how, using Restricted Boltzmann Machines, they successfully trained seven hidden layer autoencoders (neural networks whose output is trained to be as close to input as possible) at im-ages. In 2010, one of Hinton's graduate students, James Martens, developed a new Quasi-Newton approach called Hessian Free Learning, one more accurately defined as Truncated Newton Hessian Free Learning, which he says can also be used to train several hidden layers of neural networks.

5.4 Classification and Prediction of Data

Databases are full of secret knowledge that can be used to make good business decisions. Classification and prediction are two methods of data analysis that can be used to derive models representing essential groups of data or to forecast future trends in data. While classification predicts categorical marks, continuous-valued functions are predicted by models. For instance, a classification model could be constructed to predict potential customers' expenditure on computer equipment given their income and occupation. Researches in machine learning, expert systems, statistics, and neurobiology have suggested several methods of classification and prediction. Most algorithms are resident in memory, usually assuming a small size of the data. Recent mining research on the database has built on these works, creating scalable classification and prediction techniques capable of handling data from massive disc-residents. Also, these methods render computation parallel and distributed. Classification can refer to categorization, the process that recognizes differentiates and understands ideas and objects in. It's the assignment method for the predefined classes of the results. Analysis of modern systems, which is a tool for complex object analysis, is based on data mining technology as a tool for identifying frameworks and laws under not only adequate but also incomplete knowledge. Data mining algorithms mainly provide methods in the detection, classification, and clustering of objects to recreate dependencies.

Following are some of the examples of situations in which the function of data processing is called classification:

- A bank loan official wishes to examine the data to decide which customers (loan applicants) are risky or secure.

- A company's marketing manager has to evaluate a new machine to infer whether a customer with a given profile is purchasing.

- A model or classifier for predicting categorical labels is built in each of the above cases. Both marks are dangerous or secure for data related to the loan application and yes or no for marketing data.

Which is prediction?

Below are examples of cases in which the role of data analysis is called aa prediction:

- Suppose the marketing manager has to estimate how much a given consumer would spend on the business during a sale. We are bothering to predict a numeric value in this case. Therefore, the process of evaluating data is an example of numerical prediction. A model or predictor that predicts a continuous-valued-function or ordered-value will be built in this case.

Definition: Each record must contain a set of attributes, provided a series of records (training package); one of the attributes is gender. Consider a pattern for the class attribute as a function of certain attribute values.

Objective: A class should be applied as reliably as possible to documents previously not seen. To assess the model's accuracy, a test set is used. The provided data set is typically divided into training and test sets, with the training set used to create the model and test set used to validate it. If the class is numerical, the problem is a regression problem where, as opposed to a class mark, the constructed model predicts a continuous-valued function or ordered value. This is a predictive pattern. Regression analysis is a mathematical technique often used to forecast numeric.

The classification process is two-step:

1. Model: Define a set of predetermined groups. It is assumed that each tuple/sample belongs to a predefined class, as determined by the label attribute. The training set is the set of tuples that are used for model construction. The model is portrayed as laws of grouping, decision trees, or mathematical formulae.

2. Model Use: estimate the model accuracy for classifying potential or unknown objects. The identified test sample label is measured against the model's classified output. The accuracy rate is the percentage of samples that the model correctly classifies as test collection. The test set is comprised of independent of the training package. Otherwise, there would be overfitting

Prediction v/s Classification:

A bank loan officer has to examine their data to figure out which loan applicants are ―safe and who are ―risky to the bank. An All Electronics marketing manager needs data analysis to help determine whether a customer with a given profile would buy a new device. A medical researcher would like to examine the data on breast cancer to determine which of the three different therapies a patient will receive. The data analysis function in each of these examples is classification, where a model or classifier is constructed to predict categorical labels, such as −safe or −risky for loan application data; −yes or −no for marketing data; or −treatment A, −treatment B, or −treatment C for medical data. Such groups may be expressed by discrete values, where there is no sense in ordering between values. For example, the values 1, 2, and 3 may be used to describe A, B, and C treatments where there is no apparent ordering in this category of treatment regimes.

Suppose the marketing manager wants to estimate how much a given client will spend on AllElectronics during a sale. This data analysis method is an example of numerical prediction where, as opposed to a categorical mark, the constructed model predicts a continuous-evaluated function or ordered value. This is a predictor of this model. Regression analysis is a mathematical technique most commonly used for numerical prediction; thus, analogous use is often made of the two concepts. Nevertheless, we do not treat the two terms as synonyms, because as we shall see later in this chapter, many other methods can be used for numerical prediction. The two main types of predictive problems are classification and quantitative prediction. For convenience, we can use the simplified word of prediction to refer to a quantitative prediction when there is no uncertainty.

How does classification work?

The classification of data is a two-stage process. A classifier is designed in the first step, which describes a predetermined set of data classes or concepts. It is the learning stage (or training phase) in which a classification algorithm creates or builds the classifier by learning or analysing from a training set consisting of database tuples and associated class labels. A tuple- X, is represented by an n-dimensional vector attribute, X = (x1, x2,… xn), representing n-tuple measurements from n database attributes, respectively, A1, A2,::. It is assumed that each tuple, X, belongs to a predefined class as specified by another database attribute, called the class label attribute. The class mark attribute is unordered and discrete-valued. It's categorical in that each value acts as a class or category. The individual tuples which make up the training set are referred to as training tuples and are selected under review from the database. Data tuples can be referred to as tests, examples, cases, data points, or artifacts within the classification context. Since every training tuple's class label is given, this phase is also known as supervised learning (i.e., classifier learning is – supervised because it is said – How does classification work?

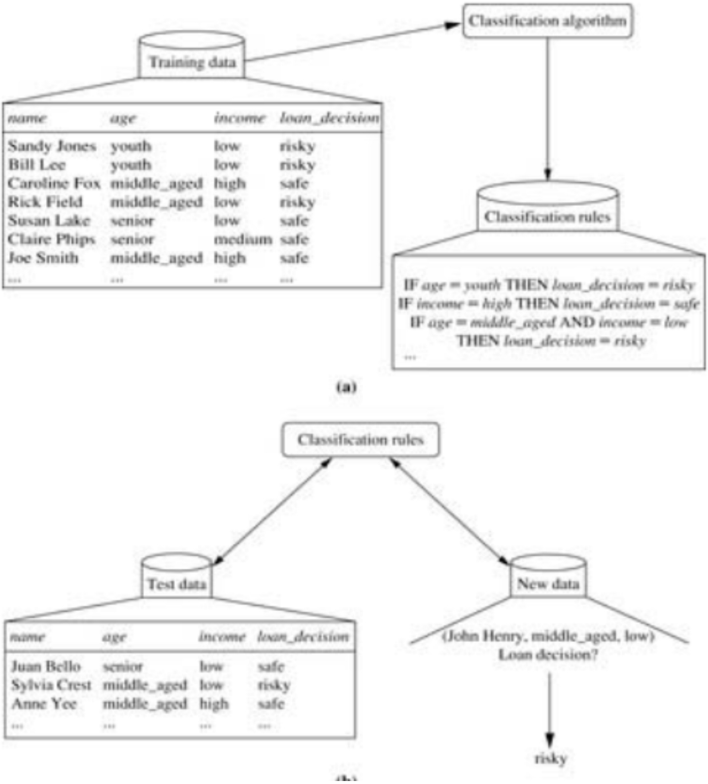

Training data
Classification algorithm
name age income loan_decision
Sandy Jones youth low risky
Bill Lee youth low risky
Caroline Fox middle_aged high safe
Rick Field middle_aged low risky
Susan Lake senior low safe
Claire Phips senior medium safe
Joe Smith middle_aged high safe
...
Classification rules
IF age = youth THEN loan_decision = risky
IF income = high THEN loan_decision = safe
IF age = middle_aged AND income = low
THEN loan_decision = risky
...
(a)
Classification rules
Test data
New data
name age income loan_decision
Juan Bello senior low safe
Sylvia Crest middle_aged low risky
Anne Yee middle_aged high safe
...
(John Henry, middle_aged, low)
Loan decision?
risky
(b)

Figure 5.18: Data Classification Process

Data classification is a two-stage process, consisting of a learning step (where a classification model is built) and a classification step (where the model is used to predict class labels for the data). The loan approval process is as shown in Figure 5.18. (Data are condensed for illustrative purposes. In practice, many more attributes can be expected to be considered. The data classification process involves:

(A) Learning: A classification algorithm analyzes the training data. The class label attribute here is loan decision, and the model or classifier being learned is defined in the form of classification rules.

(B) Classification: Test results are used to estimate classification rules accuracy. Suppose the accuracy is deemed appropriate, the rules that extend to the classification of new tuples of data.

A classifier is designed in the first step, which describes a predetermined set of data classes or concepts. It is the learning stage (or training phase) in which a classification algorithm may builds the classifier by analyzing or learning from a training set consisting of database tuples and associated class labels. A tuple - X, is represented by an n-dimensional vector attribute, representing n measurements made from n database attributes respectively on the tuple. It is assumed that each tuple, X, belongs to a predefined class as specified by another database attribute, called the class label attribute. The class mark attribute is unordered and discrete-valued. This is categorical (or nominal) in that any attribute serves as a class or category. The individual tuples which make up the training set are referred to as training tuples, and under review are collected randomly from the database. Data tuples can be referred to as tests, examples, cases, data points, or artifacts within the classification context.

Since each training tuple's class label is given, this stage is also known as supervised learning (i.e., the classifier's learning is supervised by telling a class belongs to each training tuple). This compares with unsupervised learning (or clustering), in which each training tuple's class label is uncertain, and the number of count or set of classes to be learned may not be specified beforehand. For example, if we did not have the loan decision data available for the training set, we might use clustering to try to classify groups of similar tuples, which within the loan application data that correspond to risk groups. You can also see this first step of the classification process as learning a mapping or function, which can predict the related class mark y of a given tuple X.

In this section learn a mapping or function separating the classes of data. Such mapping usually takes the form of classification codes, decision chains, or mathematical formulae. The guidelines can be used to categorize potential data tuples and provide a more in-depth insight into the quality of the data. Also, they provide a compact representation of data.

– How about the precision of classification?

Firstly, the classifier's predictive accuracy is calculated. If it is using the training set to test the accuracy of the classifier, this calculation would probably be conservative, as the classifier tends to overfit the data (i.e., during learning, it might introduce some unique anomalies of the training data that are not present in the overall data set). A test package, consisting of test tuples and their corresponding class names, is then used. These are independent of the training tuples, which means these have not been used to construct the classifier. A classifier's accuracy on a given test set is the percentage of test setup tuples that the classifier correctly classifies. The corresponding class mark for each test tuple is contrasted with the class prediction for that tuple by the trained classifier.

If the classifier's accuracy is considered appropriate, the classifier may be used to identify future tuples of data for which the class mark is not defined. (Such data are often referred to as unknown or previously unseen data in the machine learning literature).

Prediction of Data

There are several ways of data mining that are predictive. A model, for example, may forecast sales based on education and other demographic variables. Predictions have a related likelihood (How likely to be accurate is this prediction?). The probability of predictions is also known as trust (How confident can I be of this prediction?).

Some types of data mining predictive produce laws, which are conditions that indicate a given outcome. For instance, a rule or law may state that a person who has a degree and lives in a certain area or neighborhood is likely to have more than the regional average income. Rules have similar support (What percentage of the population follows the rule?).

Prediction is nothing but functions that are constantly evaluated, i.e., predicts unknown or incomplete values, by models.

Problems concerning classification and prediction

This section discusses issues with the pre-processing of the classification and prediction results. It also defines requirements for comparing and assessing classification methods.

Preparing Classification and Prediction Data

To improve the accuracy, performance, and scalability of the classification or prediction process, the following pre-processing steps can be applied to the data.

- Data cleaning: this refers to the pre-processing of data to eliminate or minimize noise (for example, by applying some smoothing techniques) and the treatment of missing values (for example, by replacing the missing value with the most frequently occurring value for that attribute or with the most likely statistical value). While most classification algorithms have some mechanisms for handling noisy or missing data, learning this step can help to reduce confusion.

- Relevance analysis: Some of the data's attributes can be redundant. Analysis of correlation can be used to assess if any two attributes given are statistically related. For example, a clear correlation between attributes A1 and A2 would indicate the removal of one of the two from further study. Even a database can have irrelevant attributes. In such instances, attribute subset selection can be used to find a reduced set of attributes such that the resulting probability distribution of the data classes is similar to the original distribution obtained using all attributes as possible. Relevance analysis in the context of correlation analysis and sub-set selection of attributes may also be used to identify attributes that do not contribute to the function of classification or prediction. The learning phase can otherwise slow down, and probably confuse, the incorporation of these attributes. Ideally, the time spent on related research will be less than the time spent learning from the resulting —reduced attribute (or feature) subset when compared to the time spent learning from the original collection of attributes. Therefore, such an analysis can help increase the efficiency and scalability of classification.

- Data transformation and reduction: Data may be transformed by normalization, particularly when neural networks or distance measurement methods are used in the learning process.

Standardization involves scaling all values for a given attribute to fall within a specified limited range, such as 1:0 to 1:0, or 0:0 to 1:0. For example, this would prevent some attributes with initially large ranges (such as, income) from weighing attributes with initially small ranges (such as binary attributes) in methods that use distance measurements.

- The data can also be translated to higher-level definitions by generalizing it. Here definition hierarchies may be found. For continuously valued qualities, this is especially useful. For example, numeric values for the income from the attribute can be generalized to discrete ranges like low, medium, and large. Likewise, categorical attributes, such as street, can be generalized to higher-level categories, such as community. Because generalization compresses the original training data, it may require less input/output operations during learning. Data can also be minimized by applying many other approaches, ranging from wavelet transformation and the study of principal components to discretionary techniques such as binning, histogram analysis, and clustering.

Comparing methods of classification and of prediction

Below are the conditions for comparing Classification and Prediction methods:

- **Accuracy** — Classifier accuracy refers to the classifier's ability to predict the class mark correctly, and predictor accuracy refers to how well a predictor may estimate the value of the expected attribute for new data.

- **Pace** - This refers to the computational expense of producing and using the predictor or classifier.

- **Robustness** - refers to the classifier's or predictor's ability to make accurate predictions of the given noisy or raw data.

- **Scalability** — Scalability points to the ability to effectively construct a classifier or predictor given the large data volume.

- **Interpretability** -It applies to the degree to which the classifier or predictor knows that.

5.5 Approaches for Data Classification

Bayesian classifiers: Bayesian classifiers are the classifiers of statistics. We can estimate the probability of class membership, such as the likelihood that a given tuple belongs to a certain class. The Bayesian classification is based on the theorem of Bayes, as described below. Studies comparing classification algorithms find that a simple Bayesian classifier known as the naive Bayesian classifier is comparable with decision tree and selected neural network classifiers inefficiency. The Bayesian classifiers also showed high precision and speed when applied to large datasets.

Rule-based classification: here, we look at classifiers based on laws, where the learned model is defined as a set of IF-THEN laws. Next, we discuss how such laws are used to define. Then we research ways they can be generated using a sequential covering algorithm, either from a decision tree or directly from the data of training.

Basics of Probability Theory

We've always thought in the logic-based methods that something is either considered to be false or to be valid. Nevertheless, it is also helpful to reflect the fact that we assume that with probability (say) 0.65, anything is possibly true, or true. It is useful for dealing with problems where there are real randomness and unpredictability in the world (such as in games of chance) and also for dealing with problems where we could figure out exactly what's true in the world if we had enough details, but where this is impractical. The definition of chance helps to have a solution.

The probability theory is the branch of probability-based mathematics, the study of random phenomena. The core objects of probability theory are random variables, stochastic processes, and events: mathematical abstractions of non-deterministic events or measurable quantities that can either be single occurrences or develop in a seemingly random fashion over time. When an actual coin toss or dice roll is called a random event, then the sequence of some random events will exhibit certain patterns, which can be observed and predicted, when replicated several times. The law of large count of numbers and the central limit theorem are two representative mathematical results that explain these patterns.

As a mathematical basis for statistics, the theory of probability is important for many human activities, which involve quantitative analysis of large data sets.

Methods of Probability

The theory also refers to representations of complex systems given only partial state-related information, as in statistical mechanics. A great discovery of physics of the twentieth century was the probabilistic existence, represented in quantum mechanics, of physical phenomena at atomic scales.

Bayes's theorem (alternatively Bayes's law or Bayes' rule) is a theorem with two distinct definitions in probability theory and statistics. In the Bayesian understanding, it explains how to rationally adjust a subjective degree of conviction to account for proof. This refers to frequentist interpretation to inverse representations of the probabilities of two cases. Bayes' theorem is fundamental to Bayesian statistics in the Bayesian understanding, and it has applications in fields including science, engineering, medicine, and law. The application of Bayes ' theorem is called Bayesian inference to update beliefs.

Introductory example: If someone informed you, they had a nice conversation on the train, the probability of being a woman they spoke to is 50%. If they told you the person, they talked to was going to attend a quilt show, it's a woman much more likely than 50%. Consider W the event "they spoke to a lady" and Q, the event "the quilt exhibition guest." Then: P(W) = 0.50; but with Q information, the modified value is P(W) which can be determined using the formula of Bayes as:

$$P(W|Q) = \frac{P(Q|W)P(W)}{P(Q)} = \frac{P(Q|W)P(W)}{P(Q|W)P(W) + P(Q|M)P(M)}$$

in which M (man) is the complement of W.

As $P(M) = P(W) = 0.5$ and $P(Q|W) >> P(Q|M)$, the updated value will be quite close to 1.

Statement and Interpretation

Bayes' theorem mathematically defines the relationship between the probabilities of A and B, P(A) and P(B) and the conditional probabilities of A given B and B given A, P(A) and P(B). In its most prevalent type, it is:

$$P(A|B) = \frac{P(B|A)\,P(A)}{P(B)}.$$

The meaning of this statement is based on the probability interpretation ascribed to the terms:

Interpretation in Bayesian Terms:

In the definition of Bayesian (or epistemological) probability tests a certain degree of confidence. Bayes' theorem also relates the degree of confidence before and after accounting for proof in a statement. Suppose, for instance; someone says a biased coin is twice as likely to land heads as tails. Originally, the degree of confidence in this maybe 50%. Afterward, the coin is flipped multiple times to gather facts. When the evidence supports the argument, confidence will rise to 70%.

For Proposition A and Proof B,

- P(A), the preceding, is the initial degree of trust in A.

- P(A|B), the degree of confidence which accounts for B, is the posterior.

- P(B|A/P(B) is the support given for in A by B.

Interpretation by Frequentists:

In frequentist interpretation, the probability of a large number of trials is specified, each resulting from a set of possible outcomes. An occurrence is a. sub-set. The probability of event A, P(A), is the percentage of trials that generate an outcome in a. Similarly, P(B) for B likelihood. When we find only experiments in which A occurs, P(B|A) is the proportion in which B occurs too. When we only find trials in which B occurs, P(A|B) is the proportion in which A occurs too. The theorem by Bayes is a fixed relation between these quantities.

For tree diagrams seen to the right, this condition can be more thoroughly visualized. The two diagrams show different ways of presenting the same facts. Suppose A, for example, has a risk factor for a medical condition, and B has the disease. Within a group, whether those with or without the risk factor are being tested depends on the proportion of the disease. The proportion that has the factor of risk depends on whether people with or without the disorder are being investigated. Such inverse representations are related by Bayes ' theorem.

Bayesian forms:

Simple form

For events A and B, provided that P(B) ≠ 0.

$$P(A|B) = \frac{P(B|A)\,P(A)}{P(B)}.$$

In a Bayesian inference step, the probability of evidence B is constant for all models A_n . The posterior may then be expressed as proportional to the numerator:

$$P(A_n|B) \propto P(B|A_n)P(A_n).$$

Extended form

Often, for some partition of the event space {A_i }, the event space is given or conceptualized in terms of P(A_i) and P(B|A_i). It is then useful to eliminate P(B) using the law of total probability:

$$P(B) = \sum_j P(B|A_j)P(A_j). \implies P(A_i|B) = \frac{P(B|A_i)\,P(A_i)}{\sum\limits_j P(B|A_j)\,P(A_j)}.$$

In the special case of a binary partition,

$$P(A|B) = \frac{P(B|A)\,P(A)}{P(B|A)P(A) + P(B|\neg A)P(\neg A)}.$$

Three or more events

Extensions to Bayes' theorem may be found for three or more events. For example, for three events, two possible tree diagrams branch in the order BCA and ABC. By repeatedly applying the definition of conditional probability:

$$P(A|B\cap C) = \frac{P(A\cap B\cap C)}{P(B\cap C)} = \frac{P(C|A\cap B)\,P(A\cap B)}{P(C|B)\,P(B)} = \frac{P(C|A\cap B)\,P(B|A)\,P(A)}{P(C|B)\,P(B)}$$

As previously, the law of total probability may be substituted for unknown marginal probabilities.

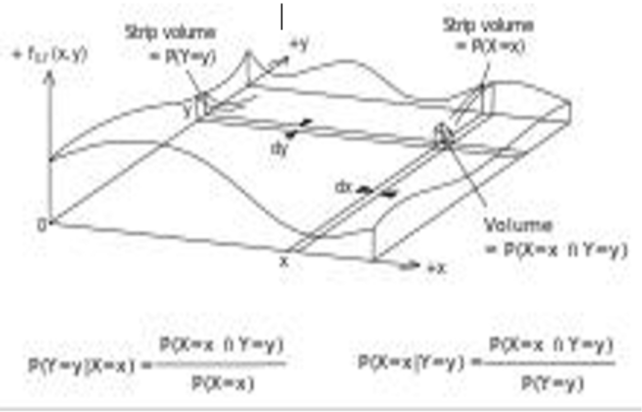


Diagram showing the sense of the theorem of Bayes as applied to an event space created by the continuous random variables X and Y; Notice that for every point in the domain there exists an instance of Bayes ' theorem. By writing the specified probability densities as a function of x and y, in reality, these instances can be parameterized.

Consider a sample space Ω where two random variables X and Y are generated. Bayes' theorem refers in theory to the cases A = {X = x} and B = {Y = y}. Terms therefore become 0 at points where each variable has finite density of probability. Bayes' theorem can be formulated in terms of the respective densities to remain useful (see Derivation).

Simple form

If X is continuous and Y is discrete,

$$f_X(x|Y=y) = \frac{P(Y=y|X=x)\,f_X(x)}{P(Y=y)}$$

If X is discrete and Y is continuous,

$$P(X=x|Y=y)=\frac{f_Y(y|X=x)\,P(X=x)}{f_Y(y)}$$

If both X and Y are continuous,

Extended form

A continuous event space is often conceptualized in terms of the numerator terms. It is then useful to eliminate the denominator using the law of total probability. For f_Y (Y), this becomes an integral:

$$f_Y(y)=\int_{-\infty}^{\infty} f_Y(y|X=\xi)\,f_X(\xi)\,d\xi.$$

Bayes' rule

Under the Bayesian interpretation of probability, Bayes' rule may be thought of as Bayes' theorem in odds form.

$$O(A_1:A_2|B)=\Lambda(A_1:A_2|B)\cdot O(A_1:A_2)$$

Where

$$\Lambda(A_1:A_2|B)=\frac{P(B|A_1)}{P(B|A_2)}.$$

Derivation of Bayes Theorem:

For general events: Bayes' theorem may be derived from the definition of conditional probability:

$$P(A|B)=\frac{P(A\cap B)}{P(B)},\ \text{if } P(B)\neq 0.$$

$$P(B|A)=\frac{P(A\cap B)}{P(A)},\ \text{if } P(A)\neq 0.$$

$$\implies P(A\cap B)=P(A|B)\,P(B)=P(B|A)\,P(A).$$

$$\implies P(A|B)=\frac{P(B|A)\,P(A)}{P(B)},\ \text{if } P(B)\neq 0.$$

For random variables: For two continuous random variables X and Y, Bayes' theorem may be analogously derived from the definition of conditional density:

$$f_X(x|Y=y) = \frac{f_{X,Y}(x,y)}{f_Y(y)} \qquad f_Y(y|X=x) = \frac{f_{X,Y}(x,y)}{f_X(x)}$$

$$\implies f_X(x|Y=y) = \frac{f_Y(y|X=x)\, f_X(x)}{f_Y(y)}.$$

5.6 Advantages and Disadvantages of Bayesian Methods

The Bayesian methods have a plenty of advantages that indicate their suitability for managing uncertainty.

To probability theory, their sound theoretical framework is the most important. But they are probably the most advanced of all the forms of reasoning for the confusion.

While Bayesian methods are more advanced than other methods of uncertainty, they are not without faults.

1. To construct a knowledge base, they need a large amount of probability data. In fact, human experts are usually unsure and insecure about the predictions they have.

2. On what prior and conditional probabilities are relevant? The sample sizes must be adequate if they are statistically dependent so that the estimates obtained are correct. If the values were given by human experts, are those values clear and comprehensive?

3. The kind of relationship between the theory and evidence is always critical in deciding how to handle the uncertainty. Reducing these comparisons to simple numbers eliminates contextual knowledge that could be required to rationalize the difficulties with performance. For example, medical diagnostic systems based on Bayes have struggled to gain acceptance because physicians mistrust systems that cannot offer explanations explaining how a conclusion was made (a function that is difficult to provide in a

system based on Bayesian).

4. The associations' reduction to numbers also removed the use of this information within other tasks. For example, the comparisons that would allow the program to justify its logic to a consumer are lost, as it is the ability to browse theories across the hierarchy of proof.

Classifier Bayesian

A Bayes classifier is one of simple probabilistic classifier based on applying a theorem of Bayes (from Bayesian statistics) with strong (naïve) assumptions of independence. A more precise word would be 'independent function model' for the underlying probability model.

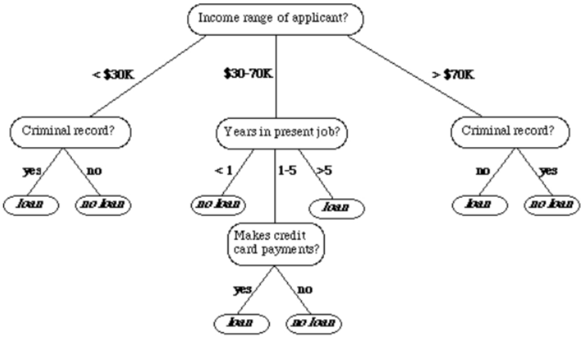
A naive Bayes classifier assumes, in simple terms, that the presence (or absence) of some particular feature of any class is unrelated to the presence (or absence) of any other feature. For example, if the fruit is red, round, and around 4 "in diameter, it can be called an apple. Even if these characteristics depend on each other or on the presence of the other characteristics, a naive Bayes classifier is considers all of these characteristics to contribute independently to the likelihood that this fruit is an apple.

Given their simplistic nature and apparently over-simplified assumptions, naive Bayes classifiers performed pretty well in many complicated real-world scenarios. The main advantage of the naive Bayes classifier is that it needs a small amount of training data to determine the parameters (means and variances of variables) required for classification.

Classification With Tree Induction Decision

Decision tree learning is a process of uses a decision tree as a predictive model that maps observations on an any item to conclusions about the target value of the item. It is one of the important predictive modeling approaches used in statistics, data mining, and machine learning. More concise names for these three models are classification trees or regression trees.

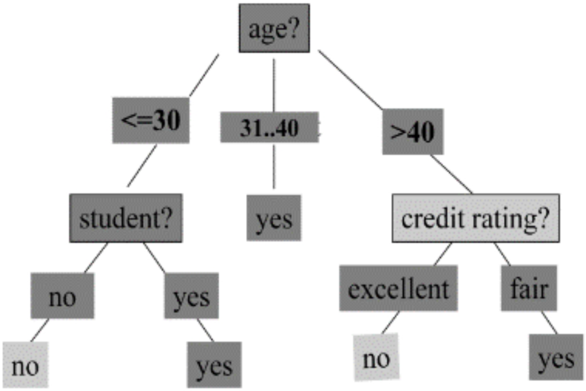
We should provide a decision tree, for example, to help a financial institution determine if an individual should be given a loan:


Income range of applicant?
< $30K
$30-70K
> $70K
Criminal record?
Years in present job?
Criminal record?
yes
no
< 1
1-5
>5
no
yes
loan
no loan
no loan
loan
loan
no loan
Makes credit card payments?
yes
no
loan
no loan

Induce a decision tree along with the decisions or classifications for certain instances from a collection of instance data.

Classification by tree-induction decision

Decision tree induction is the learning of decision trees from class-labeled training tuples. A decision tree is something like a flowchart tree structure in which each internal node (non-leaf node) denotes an attribute check, each branch represents the test outcome, and each leaf node (or terminal node) carries a class label.


age?
<=30
31..40
>40
student?
yes
credit rating?
no
yes
excellent
fair
no
yes
no
yes

A standard decision tree, as shown above. Rectangles denote the internal nodes, and ovals denote the leaf nodes. Some decision tree algorithms generate only binary trees (where each internal node branches to exactly two other nodes), while others may generate non-binary trees.

―How are classification decision trees used?

Because of a tuple, X, for which the associated class mark is unknown, the tuple attribute values are evaluated against the decision tree. A route is traced from the root to a leaf node, which holds the class prediction for that tuple. Decision trees may be easily translated to classification laws.

―Why is the tree decision classifier so popular?

Implementation of decision tree classifiers requires no domain awareness or set of criteria and is therefore ideal for exploratory information creation. Decision trees can handle high-dimensional data. Their representation of acquired knowledge in tree form is intuitive and generally easy to assimilate by humans.

Node Decision Algorithm:

Algorithm: Build a decision tree. Generate a decision tree from the data Partition D training tuple.

Input: Data partition, D, which is a set of training tuples and related class labels; a list of attributes, the set of attributes;

The attribute selection process, a technique for evaluating the dividing criterion that ―best of all divides the data tuples into individual groups. This criterion consists of a dividing attribute and likely a dividing point or dividing subset.

Methodology:

(1) Make N node;

(2) where all tuples in D are of the same class, then C

(3) return N as leaf node with Class C mark;

(4) If a list of the attributes is empty,

(5) return N as a single-class leaf-node in D; / plurality vote

(6) apply the method of selecting attributes (D, set of attributes) to find the appropriate criteria for splitting;

(7) mark node N with the criterion for splitting;

(8) If the attribute splitting is discrete and multi-way splits allowed then / is not limited to binary trees

(9) Attribute list attribute dividing attribute; / delete dividing attribute

(10) for every consequence j of the dividing criterion

- partition the tuples for each partition and expand subtrees

(11) let Dj be a data tuple set in D that meets the result j; / a partition

(12) Where Dj is empty,

(13) add to node N a leaf labeled with majority class in D;

Additionally, add the node returned to the node N by Generate decision tree(Dj, attribute list);

- Ends

(14)Return N;

Starts with some training set of tuples and their corresponding class labels. When the tree is being constructed, the training set is recursively partitioned into smaller subsets. A simple decision tree algorithm is outlined above. The technique is as follows. The algorithm is named with three parameters: D, Attribute list and Attribute selection process.

The tree begins as a single node, N, representing the training tuples in D (step 1) If the tuples in D are all of the same or realated class, then node N becomes a leaf and is labeled with that class (steps 2 and 3). Note that steps 4 and 5 are consider as terminating conditions. All terminating conditions are clarified at the end of the algorithm.

We assume the resulting partitions would be as pure as possible.

The node N is labeled with the criteria for splitting, which acts as a node check (step 7). For each of the outcomes of the dividing criteria, a branch is developed from node N. The tuples in D are subdivided accordingly (steps 10 through 11). Three situations are probable. Let A be the attribute that splits. A has v separate values, fa1, a2,::; average, based on data from the school.

1. A is discrete-evaluated: In this case, the test results at node N explicitly correspond to the known values of a. For each known value, add, of A, a branch is generated and labeled with that value. Partition Dj is the class-labeled tuple sub-set in D having meaning aj of A. If all the tuples in a given partition may have the nearly same value for A, then A doesn't need to be included in any possible tuple partitioning. And it's omitted from the list of attributes (Stages 8 through 9).

2. A is continuously evaluated: in this case, node N test has two possible outcomes, corresponding to conditions A split-point and A > split point, respectively, where the split point is the split point returned as part of the splitting criterion by the attribute selection process. (In practice, the split point A, is often taken as the midpoint of two known adjacent values of A and thus may not necessarily be a pre-existing value of point A from the training data.) These two branches are grown from N and labeled according to the above results. The tuples are partitioned in such a way that D1 holds the class-labeled tuples sub-set in D for which A split points, while D2 holds the rest.

3. A is discrete-evaluated, and a binary tree must be generated (as determined by the selection measure or algorithm used for the attribute): is the check at node N of the form −A 2 SA? ‖ . SA is the dividing subset for A, returned as part of the dividing criteria by the Attribute selection process. It is a subset of values known as A. If the value of a given tuple is A, and if aj 2

SA, then satisfied with the check at node N. Two branches are cultivated from N. By definition, yes, is defined as the left branch out of N so that D1 corresponds to the sub-set of class-labeled tuples in D that fulfill the test. The right-branch out of N is labeled no so that D2 may corresponds to the sub-set of D-labeled class tuples that do not fulfill the test. The algorithm makes recursive use of the same method to construct a decision tree for the tuples at every resulting partition, Dj, of D (step 14).

The recursive partitioning only stops if all of the following terminating conditions are true:

1. All partition D tuples (represented at node N) belong to the same class (steps 2 and 3).

2. There are no remaining attributes from which to further partition the tuples (Step4). In this case, it employs a majority voting (step 5). This includes transforming and marking node N into a leaf with the most common class in D. Alternatively; the node tuples can be stored in class distribution.

3. For a given branch, there are no tuples, that is, the partition Dj is empty (step12).

In this case, a leaf is formed in D (step 13), with the majority class.

It returns the resulting decision tree (step 15).

The computational complexity of the training set given algorithm D is O(n jDj log(jDj)), whereas n is the number of attributes describing the tuples in D, and jDj is the number of training tuples in D. This implies the computational cost of increasing a tree with jDj tuples increases at most n jDj log(jDj).

Tree Tailing

When a decision tree is created or built, many of the branches may represent anomalies due to noise or outliers in the training data. Tree pruning methods tackle this data overfitting problems. Generally, these approaches employ statistical tests to eliminate the least accurate divisions. Pruned trees tend to be a smaller and less complicated, and thus easier to grasp. Typically they are quicker and stronger than unpruned trees when properly classifying independent test data (i.e., of previously unknown tuples).

– How does pruning work for the tree?

Tree pruning has two common approaches: pre-pruning and post-pruning.

A tree is ―pruned in the pre-pruning method by stopping its construction early (e.g., by deciding not to further break or partition the training tuple subset at a given node). The node becomes a leaf after leaving. The leaf may carry the most frequent class among the tuples of subsets or the distribution of probabilities of those tuples. Measures such as statistical significance, knowledge gain, Gini index, and so on can be used when building a tree to determine a split's goodness. If partitioning the tuples at a node will lead to a split dropping below a pre-specified threshold, further partitioning of the given subset will be stopped. And there are difficulties in selecting a reasonable level. High thresholds could result in trees being oversimplified, while low thresholds could lead to very little simplification. Post-pruning is the second and more popular method, extracting sub-trees from a fully grown tree. At a given node, a subtree is pruned by removing its branches and replacing it with a leaf. The leaf is classified as having replaced the most common class among the subtree.

In the pruned tree form, the subtree in question is pruned by replacing it with the leaf ―class B. This method considers a tree's cost complexity to be a feature of the tree's number of leaves and the tree's error rate (where the error rate is the tree's percentage of tuples misclassified). It starts from the tree bottom. With each internal node, N, it calculates the sub-tree 's cost complexity at N, and the sub-tree 's cost complexity at N if pruned (i.e., substituted by a leaf node). Comparison of the two qualities. If pruning the subtree at node N will result in lower cost complexity than pruning the subtree. Otherwise, it is preserved. To measure cost complexity, a pruning set of class-labeled tuples is used. This collection is independent of the training set used to build the unpruned tree and of any test set used to estimate the accuracy.

Classification - Rule-Based

IF-THEN Rules: Rule-based classifier allows the use of the IF-THEN classification rules collection.
We can express the rule as follows:

IF Condition Or Ending

Let us find Rule R1,

R1: IF Age = Youth AND student=YES

THEN buy a computer = yes

- The IF part of the provision is known as precedence.

- The THEN part of the law is known as consequent.

- In the preceding section, the state consists of one or more tests of the attributes, and these tests are logically ANDed.

- Prediction of the class consists of the consequent portion.

Only we can write the following rule R1:

R1: (age = youth) ˆ (student = yes) (computer buys = yes))

If, for a given tuple, the condition holds the real, then the antecedent is fulfilled.

Rule Extraction

Here we'll learn how to construct a classifier based on rules by extracting IF-THEN rules from the decision tree. Mind points to excerpt rule from a decision tree:

- Every path from the root to the leaf node is generated with one law.

- Every splitting criterion from the preceding rule is logically ANDed.

- The leaf node carries the prediction of class, thus defining the law accordingly.

Using IF-THEN Classification Rules

Rules are a good way of describing bits of knowledge or facts. For classification, a rule-based classification uses a collection of IF-THEN laws. An IF-THEN law is a type of expression.
IF THEN FIND state. An example will be rule R1,
R1: IF age = youth AND student = yes machine buys = yes.
A rule's ―IF portion (or left-hand side) is known as the preceding rule or precondition. The consequent rule is the ―THEN half-part

(or right-hand side). In the preceding rule, the condition consists of one or more tests of the attributes (such as age = youth and student = yes), which are logically ANDed. The result of the rule includes a class prediction (in this case, we predict whether a client will purchase a computer). Even R1 can be written as R1: (age = youth) ˆ (student = yes)) (computer buys = yes).

If the condition (that is, all the attribute tests) in a preceding rule holds true for a given tuple, we assume the preceding rule is satisfied (or literally, the rule is satisfied), and the rule covers the tuple.

A rule R can be measured by its consistency and scope. Given a tuple, X, let ncovers be the number of tuples protected by R from a class-labeled data set, D; ncorrect is the number of tuples properly classified by R; and jDj is the number of tuples in D. We may describe R coverage and precision as

$$coverage\,(R\,) = \frac{n\ covers}{|D\,|}$$

$$accuracy\,(R\,) = \frac{n\ correct}{ncovers}$$

That is, coverage of a rule is the percentage of tuples protected by the rule (i.e., whose attribute values hold true for the precedent of the rule). They look at the tuples it protects for the consistency of a guideline, and see what proportion of them the law will properly define.

Chapter 6: Data Mining: Decision Tree Methods

Decision trees are both strong and popular for prediction and classification. The attractiveness of tree-based methods is largely due to the fact that trees represent rules of decision. Rules can be expressed easily in English so that we humans can understand them; they can also be expressed in a language for accessing databases, such as SQL, to retrieve records in a specific category. Decision trees also help to analyze data in order to gain insight into the relationships between a large number of candidate data with input variables to a target variable. Because decision trees combine both data exploration and modeling, even when building the final model using some other technique, they are a powerful first step in the modeling process.

Often there's a trade-off between model accuracy and transparency. The only thing that matters in some applications is the accuracy of classification or prediction; if a direct mail company obtains a model that can reliably predict about which members of a prospect supply are most likely to respond to a particular request, the company does not care how or why the model works. In some other cases, it is important to have the opportunity to justify the justification for a decision. For example, there are statutory prohibitions against discrimination based on such variables in insurance underwriting. The insurance provider can be in the position of having to prove to a judge that it has not used improper discriminatory tactics to issue or refuse coverage. Likewise, it is more appropriate for both the loan officer and the credit applicant to hear that a credit application was rejected on the basis of a computer-generated rule (such as income below a certain threshold and the amount of current revolving accounts above a certain another threshold) than to hear that the decision was made by a neural network generating it. This chapter begins with a review of what decision trees are, how they work, and how they can be applied to classification and problems of prediction. This then explains the core algorithm used to build decision trees and address some of the core algorithm's most common variants. Practical examples from the experience of the authors are used to demonstrate the usefulness and applicability of decision tree models and to illustrate practical considerations that need to be considered.

What Is a Tree of Decisions?

A decision tree is the structure that can be used by applying a sequence of simple rules for the decision to divide a large collection of records into successively smaller sets of records. With each successive division, the resulting sets' members are increasingly identical to each other. A good example is a familiar division of living things into kingdoms, phyla, classes, orders, families, genera, and species, invented in the 1730s by the Swedish botanist Carl Linnaeus. The phylum chordata is assigned to a single species within the animal kingdom because it has a spinal cord. Additional features are used to further subdivide the chordates into birds, mammals, reptiles, etc. These groups are further subdivided until members of the same species are not only morphologically identical at the lowest level of taxonomy; they are capable of reproducing and producing fertile offspring. A decision tree model is consisting of a set of rules to divide a large heterogeneous population into smaller, more homogeneous groups relative to a particular target variable. A decision tree can be carefully manually built in the manner of Linnaeus and the generations of taxonomists who preceded him, or it can be automatically grown by applying one of several decision tree algorithms to a collection of pre-classified data. Much of this chapter deals with the algorithms for automatically creating decision trees. Generally, the goal variable is categorical, and the decision tree model is either used to measure the probability that a given record may belongs to one of the groups, or to classify or order the record by assigning it to the most probable class. Decision trees can also be used and applied to estimate the value of a continuous variable, though other techniques are more appropriate to this task.

Classification

Anyone who is familiar with the Twenty Questions game will have no difficulty understanding how records are classified by a decision tree. In the game, one player can think of a particular place, person, or thing that all the participants would know or recognize, but the player gives no clue to his identity. Through answering a series of yes-or-no questions, the other players seek to figure out what it is. A good player rarely needs the full 20 question allotment to move all the way from "Is it bigger than a bread box? "To" the Golden Gate Bridge. "Such a series of questions represent a decision tree. Like in the game, the answer to the first question determines the question that follows. Like several members, the initial questions establish broad categories; follow-up questions split the broad categories into bigger and smaller sets. If the questions are well selected, a remarkably short series is enough to identify an incoming record accurately.

The Twenty Questions game illustrates the process of using a tree to attach a score or a class to a record. A record inputs the root node into the tree. The root node performs a check to decide which child node the record will reach next. There are various algorithms to select the initial test, but the purpose is always the same: to select the test that best discriminates between the target groups. The cycle is repeated until a leaf node is reached with the data. All records which end at a given tree leaf are classified in the same manner. Each leaf has a unique or different path from the root to it. That path is an explanation of the rule used for record classification. Different leaves that make the same classification, but each leaf for a different purpose make that classification. For instance, in a tree that classifies fruits and vegetables by color, the leaves for apple, tomato, and cherry could all predict "red," albeit with varying degrees of confidence as there are likely to be some examples of green apples, yellow tomatoes, and black cherries.

In Figure 6.1, the decision tree classifies potential catalog recipients as likely (1) or unlikely (0) to place an order if a new catalog is sent. The tree was created using the SAS Enterprise Miner Tree Viewer tool in Figure 6.1. The chart is drawn in data mining circles according to the usual convention — with the root mentioned at the top and the leaves at the bottom, maybe indicating that data miners should come out more to see how real trees grow. Every node in the upper-right corner is labeled with a node number and the expected class in the middle. The rules of decision to split each node are printed on the lines that connect each node to its children. For "lifetime orders," the division at the root node, the left branch is for customers with six or fewer orders, and the right branch is for customers with seven or more. Any record that reaches the 19, 14, 16, 17, or 18 leaf nodes are classified as likely to respond since, in this case, the class predicted is 1. The paths to those nodes of the leaf describe the tree rules. For example, if the customer has made more than 6.5 orders, and it has been less than 765 days since the last order, it is likely that the customer will react.

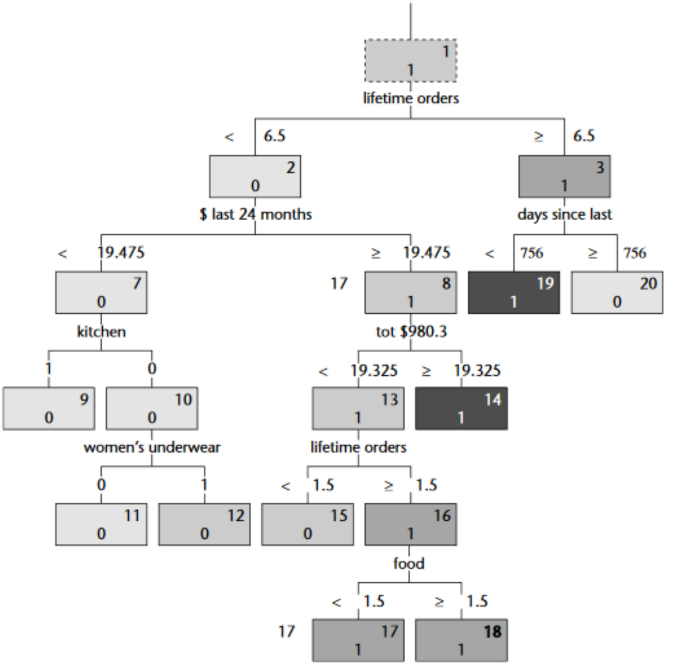

1
1
lifetime orders
<
6.5
≥
6.5
2
0
3
1
$ last 24 months
days since last
<
19.475
≥
19.475
<
756
≥
756
7
0
17
8
1
19
1
20
0
kitchen
tot $980.3
1
0
<
19.325
≥
19.325
9
0
10
0
13
1
14
1
women's underwear
lifetime orders
0
1
<
1.5
≥
1.5
11
0
12
0
15
0
16
1
food
<
1.5
≥
1.5
17
17
1
18
1

Figure 6.1 A binary decision tree classifies catalog recipients as likely or unlikely to place an order.

Warning readers may note that there seems to be no difference in some of the splits within the decision tree. For example, nodes 17 and 18 are differentiated by the number of orders they have received that included products in the category of food, but both nodes are identified as respondents. This is because although the probability of response in node 18 is higher than in node 17, it is above the threshold set for classifying a record as a respondent in both cases. The configuration as a classifier has only two outputs, one and zero. This binary classification throws away useful information that uses decision trees to produce scores and probabilities to bring us to the next topic.

Scoring

Fig 6.2 is an image of the same tree as in Figure 6.1, but using a different tree viewer and changed settings to annotate the tree with additional details — namely, the percentage of Class 1 records at each node. Now it is obvious that the tree represents a dataset that includes half responders and half non responders since the root node has a 50 percent ratio. Any node with responders greater than 50 percent is listed as "1" in Figure 6.1, including nodes 17 and 18. The discrepancy between those nodes is explained in Figure 6.2. In Node 17, 52.8 percent of the represent responders' records, whereas, in Node 18, 66.9 percent do. Clearly, a Node 18 record is more likely to represent a respondent than a Node 17 record. The proportion of records can be used as a ranking in the desired class, which is much more valuable than just the classification. A classification merely divides documents into two classes to obtain a conditional result. A score allows the records or data to be sorted from the most likely to become members of the desired class to the least likely.

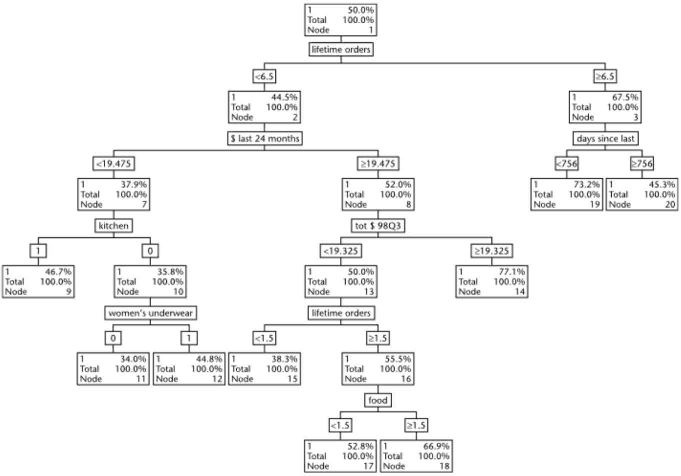

1 50.0% Total 100.0% Node 1
lifetime orders
<6.5
≥6.5
1 44.5% Total 100.0% Node 2
1 67.5% Total 100.0% Node 3
$ last 24 months
days since last
<19.475
≥19.475
<756
≥756
1 37.9% Total 100.0% Node 7
1 52.0% Total 100.0% Node 8
1 73.2% Total 100.0% Node 19
1 45.3% Total 100.0% Node 20
kitchen
tot $ 98Q3
1
0
<19.325
≥19.325
1 46.7% Total 100.0% Node 9
1 35.8% Total 100.0% Node 10
1 50.0% Total 100.0% Node 13
1 77.1% Total 100.0% Node 14
women's underwear
lifetime orders
0
1
<1.5
≥1.5
1 34.0% Total 100.0% Node 11
1 44.8% Total 100.0% Node 12
1 38.3% Total 100.0% Node 15
1 55.5% Total 100.0% Node 16
food
<1.5
≥1.5
1 52.8% Total 100.0% Node 17
1 66.9% Total 100.0% Node 18

Figure 6.2 A decision tree annotated with the proportion of records in class 1 at each node shows the probability of the classification.

A performance capable of rank-ordering a list is all it takes for several applications. It is enough to pick the top N percent for a mailing and to measure the lift at various depths in the list. Nevertheless, for some applications knowing that A is more likely to respond than B is not sufficient; we want to know the actual probability of a response from A. Assuming that a response's prior probability is known, it can be used to measure the likelihood of response from the score produced from the over-sampled data used to create the tree. Additionally, the model can be applied to classified data, which has a response distribution that represents the true population. Another technique, called back fitting, generates scores using the class proportions at the leaves of the tree to replicate the probability that a record from a similar population would be a memorial of the class.

Estimation

Suppose the relevant business question is not who is going to answer, but what is going to be the size of the next order from the client? You can use the decision tree to answer the question too. If order quantity is one of the variables in the pre-classified model collection, the average size order in each leaf can be used as the approximate order size for any unclassified record that meets the requirements for that leaf. A numeric target variable may also be used to construct the tree; that tree is called a regression tree. Rather than growing a categorical variable's purity, each split in the tree is chosen to decrease the variance within each child node in the target variable values. It's not a good idea to use the assumption that trees can (and often are) used to estimate continuous values. An estimator of a decision tree can only produce as many distinct values as the leaves in the tree. Having a continuous function is optimal for estimating a continuous variable. Regression models and models of neural networks are typically best suited for estimation.

Trees Grow in Many Forms and Pattern

The tree in Figure 6.1 is a non-uniform-depth binary tree; that is, each non-leaf node has two children, and leaves are not all at the same or equal distance from the root. In this case, each node may represent a yes-or-no question, the answer of which dissuades a record from two paths to the next tree level. Since any multiway split can be expressed as a series of binary splits, trees with higher branching factors are not in real need. However, there are several data mining tools that can generate trees of more than two branches. Many decision tree algorithms, for example, break on categorical variables by generating a branch for each class, resulting in trees with different branch numbers at different nodes. Figure 6.3 shows a tree using both three-way and two-way splits for the same question of classification as the tree in Figures 6.1 and 6.2.

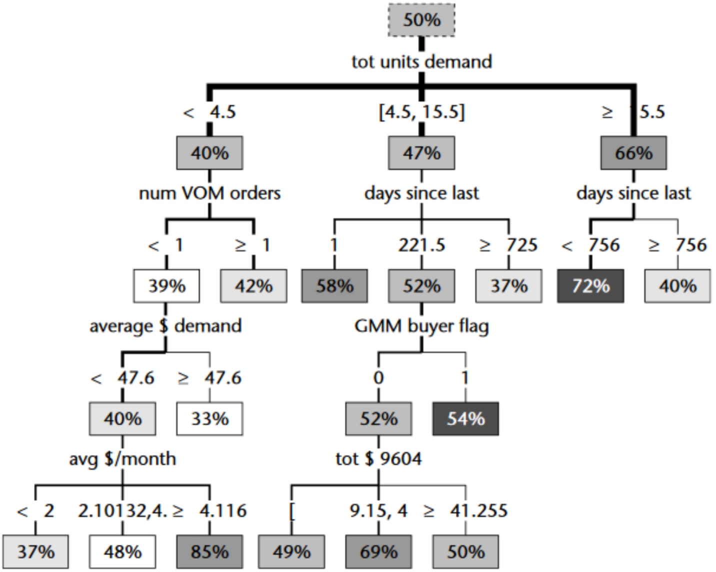

50%
tot units demand
< 4.5
[4.5, 15.5]
≥ 15.5
40%
47%
66%
num VOM orders
days since last
days since last
< 1
≥ 1
1
221.5
≥ 725
< 756
≥ 756
39%
42%
58%
52%
37%
72%
40%
average $ demand
GMM buyer flag
< 47.6
≥ 47.6
0
1
40%
33%
52%
54%
avg $/month
tot $ 9604
< 2
2.10132,4.
≥ 4.116
[
9.15, 4
≥ 41.255
37%
48%
85%
49%
69%
50%

Figure 6.3 This ternary decision tree is applied to the same the same classification problem as in Figure 6.1.

6.1 How a Decision Tree is Grown

While the core decision tree algorithm has several variants, all of them follow the same primary procedure: frequently split the data into smaller and fewer groups in such a way that each new generation group of nodes has greater purity than their ancestors regarding the target variable. We assume a binary, categorical target variable for most of this discussion, such as the answering/ nonresponse. It simplifies the definitions without losing any of their generality.

Finding the Splits

There is a training set at the outset of the process, consisting of pre-classified records — that is, the target variable value is defined for all instances. The objective is to create a tree that assigns a class (or a probability of membership in each and every class) to the target field of a new record based on the input variables values. The tree is constructed by splitting the records according to a function of a single input field at each node. Then the first step is to determine which of the fields of input does the best split. The best split is described as one that does the best job of splitting the records into groups where each group has a single class predominating. The criterion used to test a possible divide is purity. The next section of deals in more detail with specific methods for calculating purity. But they're all trying to get the same effect. Low purity means the set includes a representative class distribution (relative to the parent node) of all of them, whereas high purity means that members of a single class predominate. The better split is that which raises by the largest amount the integrity of the record sets. A successful split often generates nodes of similar size or does not create nodes that have very few records, at least. Such thoughts are clearly simple to see.

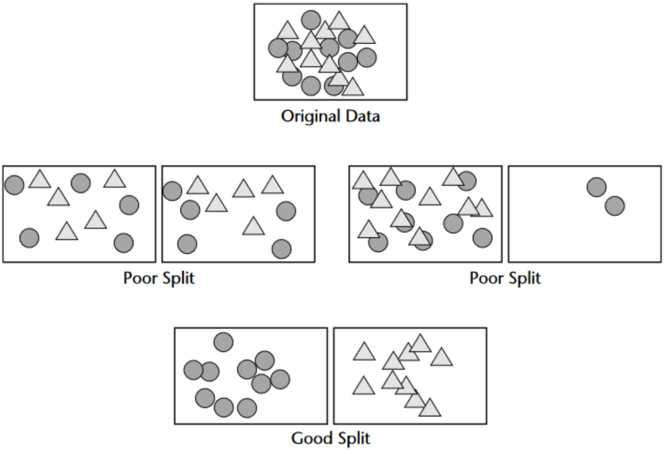


Figure 6.4 A good split increases purity for all the children.

The first split is a bad one since the purity does not increase. The initial population includes equivalent numbers of the two kinds of dots; any child does likewise after the break. The second split is also bad since even though purity is significantly improved, the pure node has few members, and the larger child's purity is only marginally better than the parent's one. The final split is a good one because it leads to children of approximately the same size and far greater purity than the parent. The algorithms for building trees are exhaustive. We continue by in turn-taking each input variable and calculating the increased purity resulting from each split indicated by that variable. After all the input variables have been tried, the one which yields the best split is used for the initial split, creating two or more children. If no split is feasible (because too few records are available) or if no split makes a change, then the algorithm ends with that node, and the node becomes a leaf node. Otherwise, the algorithm will perform the split and repeat on each of the kids. An algorithm that thus repeats itself is called a recursive algorithm. Splits are assessed in terms of the target variable, based on their effect on node purity. This implies that selecting a suitable criterion for Splitting depends on the type of target variable, not the type of input variable. A test such as Gini, knowledge gain, or chi-square is acceptable with a categorical target variable if the input variable which provides the split is numerical or categorical. Likewise, a method such as a variance reduction or the F-test is ideal for measuring the split with a continuous quantitative variable regardless of whether the input variable supplying the split is categorical or numerical.

Splitting on a numeric input Variable data

If looking for a binary split on a numerical input variable, each value that the variable assumes in the training set is viewed as a split candidate. Splits take the form $X < N$ on a numeric scale. All records are sent to one child where the value of X (the splitting variable) is less than some constant N, and all records where the value of X is greater than or equal to N are sent to the other. Regardless of the split, the change in purity is measured, if any, after each trial break. Many implementations of the splitting algorithm do not directly evaluate every value in the interests of efficiency; rather, they evaluate a subset of the values. The only use that it makes of numeric inputs when the decision tree is graded is to compare its values with the split points. They are never combined or added together by weights like they are in many other versions. It has the significant implication that the decision trees are not prone to outliers or distorted distributions of numeric variables, since the tree uses only the rank of numeric variables and not their absolute values.

Splitting on a Categorical Input Variable

For splitting on a categorical input variable, the simplest algorithm is simply to create a new branch for each class that the categorical variable will take over. And if color is chosen as the best field on which to break the root node, and the training collection contains records that take on the values red, orange, yellow, green, blue, indigo, and violet, then the next tree stage will have seven nodes. Several software packages do use this strategy, but it also yields bad performance. High branching factors are increasingly growing the population of training records available in lower tree rates at each node, making more splits less effective. A more popular approach is to group together the classes which predict similar outcomes, taken individually. More specifically, if two classes of the output variable classes yield distributions that do not differ substantially from each other, then the two classes may be combined. The chi-square test is the standard method for determining if the distributions vary significantly.

Splitting in the presence of Missing Values

Some of the nicest aspects of the decision trees are their ability to manage missing values in either numerical or categorical input fields by actually treating null as a possible value for their own branch. This strategy is superior to tossing out missing-value records or attempting to impute missing values. Dropping out records due to missing values is likely to produce a skewed training collection because records with missing values are unlikely to be a random sample population. Replacing missing values with imputed values carries the risk that the model may disregard crucial information given by the lack of value. We have seen several situations where there is a predictive value to the fact that a particulate value is null. In one scenario, the count of non-null values in the appended demographic data at the household level was positively associated with the answer to a term life insurance bid. Apparently, people who leave more traces in Acxiom's household database (by purchasing homes, getting married, registering goods, and subscribing to magazines) are more likely to be involved in life insurance than those whose lives leave more fields null.

While it is always very useful to break on null as a separate class, at least one data mining tool provides an alternative approach too. Each node in Enterprise Miner stores many possible rules for splitting, each based on a different field of data. The algorithm uses the surrogate split based on the next best available input variable if a null value is found in the field that generates the best splits.

Growing the Full Tree

The initial split generates two or more child nodes, each subsequently split in the same way as the root node. Yet again, all fields of data are known as splitters of candidates, including fields already used for splits. Nevertheless, fields that only assume one value are excluded from consideration because there is no way they can be used to build a break. A categorical field used higher up in the tree as a splitter is likely to become relatively easily single-valued. The best split is determined for each of the remaining fields. If no split can be found that dramatically increases the integrity of a given node, or when the number of records in the node reaches some predetermined lower bound, or when the tree depth reaches some predetermined maximum, the search for that branch is discontinued and the node is classified as a leaf node.

Finally, no further splits can be found anywhere in the tree, and the entire decision tree has been grown. As we can see, usually, this complete tree is not the tree that does the best job of classifying a new collection of records. Decision-tree-building algorithms start by trying to find the input variable that will do the best job of splitting the data among the categories you want. The subsets generated by the preceding split are themselves split at each successive tree-level according to whatever rule works best for them. The tree continues to grow until better ways of dividing incoming records can no longer be found. If a totally deterministic relationship existed between the input variables and the target, this recursive splitting would ultimately produce a tree with pure leaves.

Things of this type are easy to produce, but they do not happen too much in marketing or CRM applications. Data on consumer actions hardly includes these simple, deterministic relationships between inputs and output. The fact that in terms of the available input variables, two customers have the exact same description does not ensure that they will exhibit the same behavior. A decision tree for a catalog response model could include a leaf representing women over 50 years of age, three or more last year purchases, and a total lifetime expenditure of more than $145. Typically, the customers who reach this leaf will be a mix of respondents and nonresponses.

When the leaf in question is called "responder," then the rate of error for this leaf is the proportion of nonrespondents. The lift at this leaf is the ratio of the proportion of responders in this leaf to the proportion of respondents in the population. One situation where deterministic rule is likely to be discovered is when data trends represent the rules of the market. The writers had brought the point home to them through an internship at Caterpillar, a diesel engine manufacturer. We developed a tree model of a decision to predict which warranty claims will be accepted. At the time, the company had a program that automatically paid for such claims. The findings were startling: On unknown test data, the model was 100 percent accurate. In other terms, Caterpillar had learned the exact rules it used to identify the statements. A neural network method on this issue has been less effective. For example, it may not be especially helpful to discover proven business rules; it does, however, underline the usefulness of decision trees on rule-oriented issues. Most fields, ranging from genetics to industrial processes, do actually have underlying laws, although noisy data can make them very complicated and obscure. Decision trees are a normal alternative when you believe fundamental laws are in nature.

Measuring the Effectiveness of Decision Tree

The usefulness of a decision tree, taken in its entirety, is measured by applying it to the test set — a series of records not used to construct the tree — and by evaluating the percentage correctly graded. This provides the tree as a whole classification error rate, but it is also important to pay attention to the quality of the tree's individual branches. Each path through the tree is a law, and some rules are better than others.

We can calculate, at each node, is a leaf node or a branch node:

- Number of records that reach the node
- Percentage of records in every class
- How would those records be classified if the node was a leaf?
- Percentage of documents correctly listed on this node
- The variance in the distribution between the training set and the test set.

The percentage of records correctly classified at that node is of particular interest. Surprisingly a higher node in the tree often does a better job of classifying the test set than lower nodes.

Tests for Choosing Best Split Choices

There are variety and number of different methods available for determining potential splits. Algorithms developed in the machine learning community concentrate on increasing the purity resulting from a split, whereas those developed in the community statistics concentrate on the statistical significance of the difference between the child node distributions. Alternative criteria for splitting frequently lead to trees that look very different from each other but function similarly. That is because usually there are many splits of candidates with very similar performance. Different purity measures lead to the selection of different candidates, but since all the measures attempt to capture the same corresponding idea, the resulting models tend to behave likewise.

Purity and Diversity

The first edition of this book defined requirements for splitting in terms of declining diversity resulting from splitting. In this edition, instead, we're referring to the increase in purity, which appears a little more intuitive. The two sentences allude to the same idea. A measure of purity ranging from 0 (when no two items in the sample are in the same class) to 1 (when all items in the sample are of the same class) can be converted into a measure of diversity by subtracting it from 1. Some of the measures used to assess decision tree split assign a pure node to the lowest score; others assign a pure node to the highest score. This discussion refers to all of them as a measure of purity, and the goal is to maximize purity by minimizing or optimizing the measure chosen. Figure 6.5 indicates the break well. The parent node includes identical light numbers and dark dots. The left child has nine light-points and one dark line. The right child has nine dark lines and a red line. The purity has obviously increased, but how do we measure the increase? So how does one equate this split to others? It includes a formal definition of purity, of which a number are mentioned below.

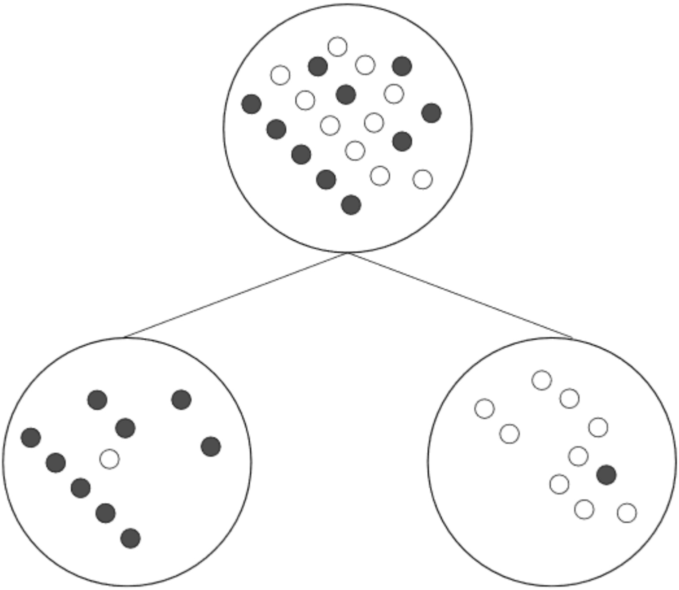


Figure 6.5 A good split on a binary categorical variable increases purity.

Purity steps for the measurement of splits for categorical target variables include:

- Gini (also known as demographic diversity)
- Entropy (also known as acquiring information)
- Gaining information ratio
- Chi-square test

If the target variable is numeric, one approach is to bin the value and use one of the above-mentioned measures. Yet there are two widely used steps for numerical targets:

- Reduction in variance
- F - Test

Note that choosing a proper measure of purity depends on whether the target variable is categorical or numeric. The type of the input variable doesn't matter, so a whole tree is constructed with the same measure of purity. A numerical input variable (AGE > 46) or a categorical variable (STATE is a member of CT, MA, ME, NH, RI, VT) could provide the split illustrated in 6.5. The children's innocence remains the same no matter what sort of break.

Gini or the Diversity of Population

One common criterion of splitting is called Gini, after Corrado Gini, an Italian statistician, and economist. This calculation, which is also used by biologists and ecologists who research population diversity, gives the likelihood that two things randomly picked from the same population are in the same class. For a population that is pure, this probability is 1. A node's Gini measure is simply the sum of the squares of class proportions. The parent population has an equal number or amount of light and dark dots for the division shown in Figure 6.5. A node with equal numbers for each of two classes has a score or number of 0.52 + 0.52 = 0.5, that is expected and required because one out of two is the chance to pick the same class twice by random selection with replacement. For all of the resulting nodes the Gini score is 0.12 + 0.92 = 0.82. A perfectly pure node would score to Gini 1. A node similarly balanced will get a Gini score of 0.5. Sometimes the scores are doubled, and then 1 is subtracted, so 0 to 1.

Nevertheless, such a modification does not make the difference to maximize purity when comparing different scores. To measure a split's effect, take each child's Gini score and multiply it by the proportion of records that enter that node and then add up the resulting numbers. In this case, since the records are divided equally between the two nodes that emerge from the divided, and each node has the same Gini score, the score for the split is the same as for one of the two nodes.

6.2 Entropy Reduction or Information Gaining

Information uses a clever idea to define pureness. When a leaf is fully pure, then it is simple to define the groups in the leaf — they all fall into the same class. However, if a leaf is really impure, then it is far more difficult to describe it. Information theory, a part of CS (computer science), has invented a metric called entropy for this case. To information theory, entropy is the measure of how disorganized a system is. This book is far beyond the scope of a detailed introduction to information theory. The intuitive notion for our purposes is that the number of bits needed to describe a given situation or outcome depends on the size of the set of possible outcomes. Entropy can be used as a calculation of the number of yes / no questions that will be needed to assess the state of the system. When there are 16 possible statements, log2(16) or four bits are required to enumerate them or describe a particular state. Additive information reduces the number of questions required to determine the state of the system, so gaining information means the same thing as reducing entropy. Both terms are used for describing algorithms in the decision tree. The entropy of a given decision tree node is the sum of the proportion of records relating to a particular class, multiplied by the base two logarithms of that proportion, overall, the classes contained in the node. (This sum is typically multiplied by – 1 to produce a positive number) The entropy of a split is just the sum of the entropies of all the nodes resulting from the split weighted by the proportion of each node's records. The algorithm searches for the split that may reduce or decrease entropy (or, equivalently, increases information) by the greatest amount when entropy reduction is chosen as a splitting criterion. For a binary target variable like the one shown in Figure 6.5, a single node entropy formula is

$$-1 * (P(dark)\log_2 P(dark) + P(light)\log_2 P(light))$$

In this case, both P(dark) and P(light) are one-half. Plugging 0.5 into the entropy formula indicates:

$$-1 * (0.5 \log_2 (0.5) + 0.5 \log_2 (0.5))$$

The first term refers to the light dots and the second term refers to the dark dots, but since there are equal numbers of light and dark dots, the term refers to -1 * log2 (0.5) which is +1. What is the entropy resulting from the split between nodes? It has its dark dot and nine-light dots, and the other has nine dark dots and one light dot. Obviously, they each have the same entropy level. In other terms:

$-1 * (0.1 \log_2 (0.1) + 0.9 \log_2 (0.9)) = 0.33 + 0.14 = 0.47$

Multiply the entropy of each and every node by the proportion of records that enter that node to measure the overall entropy of the network after the break, and sum them together to get an average. Each of the new nodes in this example receives half the information, so the total entropy is the same or equal as the entropy of each of the nodes, 0.47. Therefore, the total entropy reduction or gain in information due to the split is 0.53. It is the statistic used to equate the break with other candidates.

Information Gain Ratio

Combined with a dividing technique that handles categorical input variables by establishing a separate branch for each value, the entropy split measure will run into trouble. This was the case with ID3, a method developed by Australian researcher J for the decision tree. Ross Quinlan became part of many commercial data mining software packages in the nineteen-eighties. The problem is that the number of classes represented in each node continues to go down, and with it, the entropy, simply by splitting the larger data set into several small subsets.

The decrease in entropy due solely and individually to the number of branches is termed a split's intrinsic information. (Remember that entropy is defined as the sum over all branches of the probability of each and every branch time the log base 2 of that probability. For a random n-way split, the possibility of each branch is like 1 / n. Therefore, entropy due solely to splitting from the n-way split is simply n * 1 / n log (1 / n) or log (1 / n).

Due to the intrinsic information of multi-way splits, decision trees constructed using the criterion of entropy reduction splitting without any correction for the intrinsic information due to the split tend to be very bushy. Bushy trees with lots of multi-way splits are unwanted because these splits lead to small numbers of records in each node, a formula for unstable models. In response to this problem, C5 and other ID3 descendants who used information gain now use the ratio of the total data or information gain due to a propositioned split to the intrinsic information or data attributable solely to the number of branches created as the criterion for the evaluation of the proposed splits. This test decreases the inclination towards very bushy trees, which was an issue in the software packages for earlier decision trees.

Chi-Square Test

The chi-square (X2) test is a statistically significant chance test developed in 1900 by the English statistician Karl Pearson. Chi-square is known as the sum of the squares of the standardized differences between the predicted and observed frequencies between multiple disjoint samples of some occurrences. In other words, the test or analysis is a measure of the likelihood that the only chance creates an observed difference between the samples. Using higher chi-square values when used to measure the purity of decision tree splits means that the variation is more significant, and not merely due to chance.

For example, assume the target variable is a binary flag indicating whether or not at the end of the promotional offer duration consumers continued their subscriptions, and the proposed split is on the acquisition path, a categorical variable with three classes: direct mail, outbound call, and email. Unless the acquisition channel had no impact on the renewal rate, the number of renewals in each class would be assumed to be equal to the number of customers acquired via that process. The chi-square test subtracts the number of renewals predicted from the actual renewals observed for each source, squares the difference, and divides the difference by the number expected. To complete the score, the values for each class are added together. The score is adequate to calculate the integrity of a split in a decision-tree. A high score means the planned division divides the population effectively into subpopulations with substantially different distributions. The chi-square test or analysis gives its name to CHAID, a well-known algorithm in the decision tree first published in 1975 by John A. Hartigan. The main acronym stands for Automatic Interaction Detector Chi-square.

The original motive for CHAID was to detect statistical interactions between variables, as indicated by the term "automatic interaction detector" This does so by creating a decision tree, and the approach has also come to be used as a classification tool. CHAID uses the Chi-square test in several ways — first to merge classes that don't have significantly different or various effects on the target variable; then to choose the best split, and finally to decide if additional splits on a node are worthwhile. The new trend in the research community is away from approaches that just tend to break as long as it seems likely to be useful and towards methods involving pruning. Nevertheless, some researchers also favor the initial CHAID method, which isn't based on pruning. The chi-square method applies to categorical variables, such that the input variables must be categorical in the classic CHAID algorithm. Continuous variables need to be binned or substituted with ordinal classes such as high, medium, low. Many Current Decision Tree applications like SAS Enterprise Miner use the chi-square test to create splits using categorical variables, but use another F test to create splits on continuous variables.

F Test

The F check, named for another famous Englishman — statistician, astronomer, and geneticist, Ronald A. Fisher. — it is another split criterion that can be used for numeric target variables, despite, or maybe because of, the large difference in their areas of interest, Fisher and Pearson apparently didn't get along. Fisher's test does what Pearson's chi-square test does for categorical variables with continuous variables. It provides a measurement of the probability that samples with different means and variances are actually drawn from the same population. There is a well-understood relationship between the variance of a sample and the variance of the population from which it was taken. (In general, as long as the samples are of a fair size and randomly drawn from the population, the sample variance is a decent approximation of the population variance; very small samples — with fewer than 30 measurements — generally have a higher variance than their respective populations.)

The F test tests the relationship between two population variance estimates — one drawn from the population variance. If the separate samples are taken randomly from the same population, then these two estimates will be in close agreement. The F score reflects the average of the two figures. It is determined by the pooled sample estimate dividing the between-sample estimate. The higher the score, the less likely it is that the samples are all randomly selected from the same population. In the sense of the decision tree, a high F-score implies that the population has been effectively divided into subpopulations of substantially different distributions by a proposed division.

Pruning

As mentioned above, the decision tree continues to develop as long as new splits can be found, which enhances the tree's ability to divide the training records set into increasingly pure subsets. Such a tree has been designed for the training set, and removing any leaves on the training set will only increase the tree's error rate. Does this mean the complete tree would do the best job of classifying new data sets as well? Definitely not! A decision-tree algorithm first makes its best break at the root node where there is a large record population. As the nodes get smaller, the mechanism is dominated by idiosyncrasies of the specific training records at a node. One of the ways to think of this is that the tree seeks general patterns unique to the training set in the small nodes at the big nodes and patterns; that is, the tree overfits the training set. The result is an unstable tree that makes no good predictions. The remedy is to remove unstable splits by combining smaller leaves through a method called pruning, a thorough overview of three general approaches to pruning.

The algorithm of CART Pruning

CART is a common tree decision algorithm that was first published in 1984 by Leo Breiman, Jerome Friedman, Richard Olshen, and Charles Stone. The acronym stands for Regression and Classification Trees. The CART algorithm grows binary trees and continues to break, as long as new splits that increase purity can be found. As shown in Fig 6.6, there are several simpler subtrees within a complex tree, each reflecting a different trade-off between the complexity of the model and the rate of misclassification set by the training. A set of these subtrees is defined as candidate models by the CART algorithm. These subtrees of candidates are added to the validation set, and the tree with the lowest validation set rate of misclassification is selected as the final model.

Creating the Candidates Subtrees

The CART algorithm distinguishes subtrees of the candidates via a repeated pruning cycle. The intention is to prune those branches first with the least extra predictive power per leaf. CART relies on a principle called the modified error rate, in order to classify these least useful branches. This is a measure that raises the rate of misclassification of each node over the training set by introducing a penalty for complexity dependent on the number of leaves in the tree. The modified error rate is used to classify and label poor branches (those whose misclassification rate is not sufficiently low to remove the penalty).

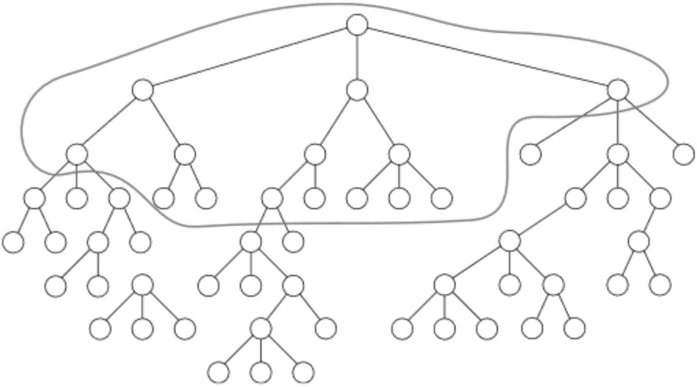


Figure 6.6 Inside a complex tree, there are simpler, more stable trees.

The formula for the modified error rate is

$$\mathbf{AE(T) = E(T) + \alpha leaf\_count(T)}$$

Where α is a gradually-increased adjustment factor for generating new subtrees. If α is zero, the modified rate of error is equal to the rate of error. The modified error levels for all possible subtrees containing the root node are evaluated as α is increasingly increased to find the first subtree. When any subtree's adjusted error rate is less than or equal to the adjusted error rate for the entire tree, we've found the first subtree candidate, α1. All branches not part of α1 are pruned, and the cycle begins again. To build a tree α2, the α1 tree is pruned. The cycle ends when the tree is pruned right down to the root node. Each of the subtrees resulting (sometimes called the alphas) is a final model candidate. Notice that all the candidates have the root node, and the biggest candidate is the whole tree.

Picking the Best Subtree

The further step is to pick the one that works best on new data from the pool of candidate subtrees. That is, of course, the validation set's intent. -- subtree for the candidate is used to identify the records in the validation package. The winner is declared to be the tree that performs this task with the lowest overall error rate. The winning subtree was sufficiently pruned to eliminate the effects of overtraining but not so much as to lose significant informative interest. The graph in Figure 6.7 shows the impact pruning has on the accuracy of classification. In more depth, the technical aside goes into this. Since this pruning algorithm is based solely on the rate of misclassification, without taking into consideration the likelihood of each classification, it replaces any subtree whose leaves all make the same classification with a common parent, which also makes that classification. In applications where the goal is to select a small proportion of the records (for example, the top 1 percent or 10 percent), this pruning algorithm may harm the tree's efficiency, as some of the leaves removed contain a very high proportion of the target class. Some tools, like SAS Enterprise Miner, allow the user to prune trees for these situations in an optimized way.

Using the Test Set to Evaluate the Final Tree

When applied to the task of classifying the records in the validation collection, the winning subtree was selected based on its total error rate. But while we expect the selected subtree to continue to be the best performing subtree when applied to other datasets, its effectiveness may be slightly overstated by the rate of error that caused it to be picked. Significant numbers of sub-trees are likely to be performed by just about as well as the one is chosen. To some degree, with that particular collection of records, the one that provided the lowest error rate on the validation set might simply have "gotten lucky" The selected subtree shall be added to a third pre-classified dataset which is disconnected from both the validation set and the train-set. This third set of data is called the test set. The error rate obtained on the test set is used to predict the expected performance of the tree-represented classification rules when applied to unclassified data.

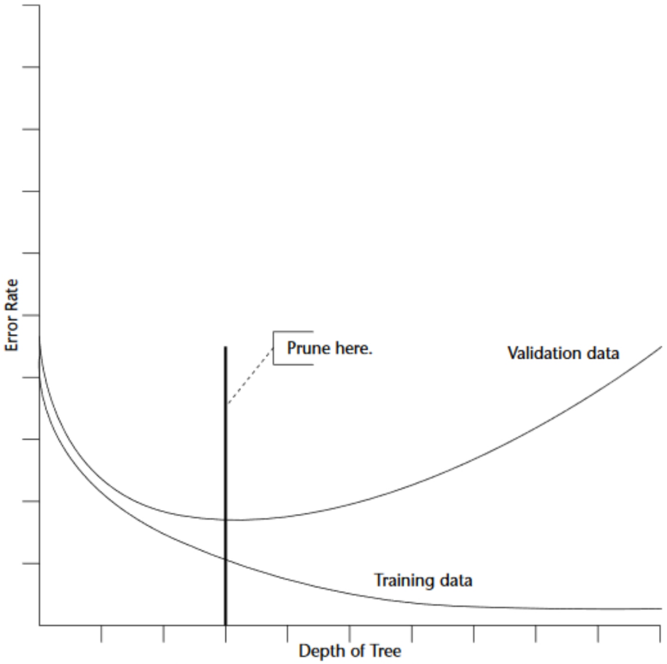


Figure 6.7 Pruning chooses the tree whose miscalculation rate is minimized on the validation set.

C5 Pruning Algorithm

C5 is the new edition of Australian researcher's decision-tree algorithm, J. Ross Quinlan has a long history of developing and refining. In the field of machine learning, an earlier edition, ID3, published in 1986, was very popular, and its successors are used in many commercial data-mining products. (The name ID3 stands for 'Iterative Dichotomiser 3.' We haven't heard a reason for the name C5, but we can infer that the context of Professor Quinlan is mathematics rather than marketing.) C5 is available as a RuleQuest commercial product (www.rulequest.com).

The trees grown by C5 are similar to those grown by CART (though C5 allows multiway splits on categorical variables, unlike CART). The C5 algorithm, like CART, first grows an overfit tree and then prunes it back to form a more stable model. But the pruning technique is very different. C5 does not use a validation collection to choose from among the candidate subtrees; it also uses the same data used to grow the tree to determine how the tree should be pruned. This may reflect the roots of the algorithm in the academic world, where in the past, it was difficult for university researchers to get their hands-on large amounts of real data to be used for training sets. As a result, they've spent a lot of time and money trying to squeeze the last few drops of information from their depleted datasets — a challenge that business world data miners don't face.

Pessimistic Pruning

C5 prunes the tree by evaluating the rate of error at each node and assuming the true rate of error is considerably worse. If N records appear at a node, and E of them is incorrectly labeled, then the error measure rate at that node is E / N. The whole point of the tree-growing algorithm now is to minimize this rate of error, so the algorithm assumes that E / N is the best it can do. C5 uses a statistical sampling comparison to arrive at an estimation of the worst rate of error likely to be seen on a leaf.

The analogy that works by thinking of the data at the leaf as reflecting the results of a set of trials, each of which can generate one of two possible outcomes. (Heads or tails is the typical example) Statisticians have been researching this particular situation since at least 1713, the year in which the famous binomial formula of Jacques Bernoulli was posthumously published. And there are well-known formulas to decide what it means to have observed E occurrences in N trials of any case. In particular, there is a formula that gives the confidence interval — the range of expected E values — for a given confidence level. C5 assumes that the number of errors found on the training data is the low end of this range, and replaces the high end to get the expected error rate of a leaf, E / N on unseen data. The smaller the node, the higher the rate of errors. If the high-end estimate of a node's number of errors is less than the estimate for its children's mistakes, then the kids are pruned.

Stability-Based Pruning

There is an issue with the pruning algorithms used by CART and C5 (and indeed with all the commercial decision tree methods used by the authors). They fail to prune those obviously unstable nodes. One good example is the break shown in Figure 6.8. The image was created by SAS Enterprise Miner for displaying a tree using its default settings. The numbers on each node's left-hand side display what's going on at the training set. The numbers on each node's right-hand side display what's going on in the validation package—trying to recognize churners, this particular tree. The highlighted branch seems to be doing very well when only the training data are taken into consideration; the concentration of churners increases from 58.0 percent to 70.9 percent. Unfortunately, the concentration of churners actually decreases from 56.6 percent to 52 percent when the very same rule applies to the validation collection. One of the model's principal objectives is to make accurate assumptions on previously unseen data.

Any rule that cannot attain the objective should be excluded from the model. A lot of data mining tools require the user to manually prune a decision tree. This is a valuable and useful facility, but we look forward to an optional data-mining program that offers automated pruning based on stability. This software will require a less arbitrary criterion to reject a split than "the distribution of the results of the validation set looks different from the distribution of the results of the training set." One alternative would be to use a statistical significance measure, such as the chi-square test or the difference in proportions. If the trust level is less than any user-defined threshold, the split will be pruned, so only splits that are, say, 99 percent positive about the validation set will remain.

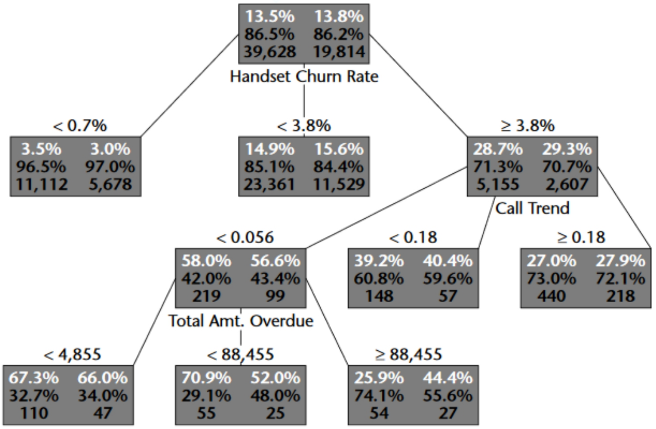

13.5% 13.8%
86.5% 86.2%
39,628 19,814
Handset Churn Rate
< 0.7%
3.5% 3.0%
96.5% 97.0%
11,112 5,678
< 3.8%
14.9% 15.6%
85.1% 84.4%
23,361 11,529
≥ 3.8%
28.7% 29.3%
71.3% 70.7%
5,155 2,607
Call Trend
< 0.056
58.0% 56.6%
42.0% 43.4%
219 99
< 0.18
39.2% 40.4%
60.8% 59.6%
148 57
≥ 0.18
27.0% 27.9%
73.0% 72.1%
440 218
Total Amt. Overdue
< 4,855
67.3% 66.0%
32.7% 34.0%
110 47
< 88,455
70.9% 52.0%
29.1% 48.0%
55 25
≥ 88,455
25.9% 44.4%
74.1% 55.6%
54 27

Figure 6.8 An unstable split produces very different distributions on the training and validation sets.

Rules of Extraction from Trees

If a decision tree is mainly used to produce scores, it's easy to forget that a decision tree is, in fact, a set of rules. If one of the aims of the data mining effort is to gain an understanding of the problem domain, reducing the enormous tangle of rules in a decision tree to a smaller, more comprehensible set might be helpful. Many cases are where the desired output is a set of rules. We define the application of decision trees to an industrial process improvement problem in Mastering Data Mining, namely the prevention of any sort of printing defect. In that case, the data mining project's end result was a small list of basic rules which could be rather placed on the wall next to each click. When a decision tree is used to produce scores, it is desirable to have a large number of leaves as each leaf produces a different score—the lesser rules, the better, when the goal is to create laws. Fortunately, the collapse of a large tree into a simpler set of laws is always possible. The first step in that specific direction is to merge tracks leading to leaves making the same classification. In Figure 6.9, the partial decision tree provides the following rules:

- See the game and home team wins and out with friends then beer.
- See the game and home team wins and sitting at home then diet soda.

- See the game and home team loses and out with friends then beer.
- See the game and home team loses and sitting at home then milk.

You can merge the two laws that forecast beer by dropping the check for whether the home team wins or loses. This test is necessary to differentiate between milk and diet soda, but it has no effect on the consumption of beer. The new, simpler rule is
Watch the game and out with friends then beer.

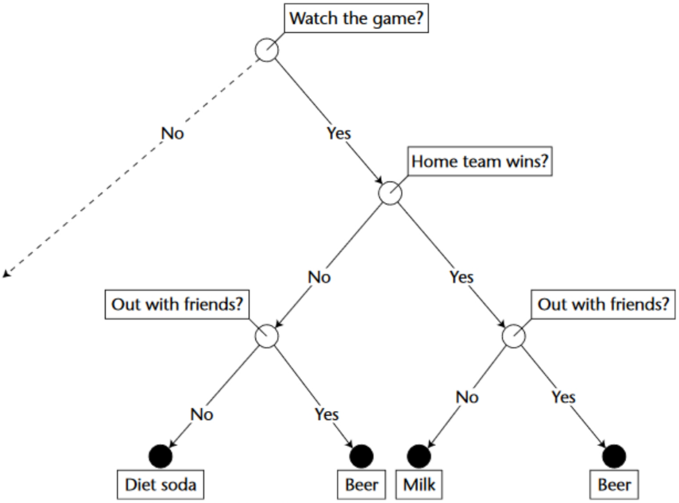


Figure 6.9 Multiple paths lead to the same conclusion.

Nothing is controversial up to this point, since no information was lost, but the rule generator for C5 goes further. It tries to generalize each rule by eliminating clauses, then comparing the expected error rate of the new, briefer rule with that of the original one using the same pessimistic error rate assumption first used to prune the tree. The rules for several different leaves are also applied to the same law, and this method results in fewer rules than the leaves that the decision tree had. The growing record ends up at exactly one leaf in the decision tree, so each record has a definite classification.

However, after the rule-generalization process, rules that exist which are not mutually exclusive and records which are not cogenerated by any rule. The first query can be solved by simply choosing one rule when more than one is applicable. The second query or issue involves the implementation of a class allocated by default to any record not protected by any of the rules. The most commonly occurring class is usually selected as the norm. If a set of generalized rules has been developed, Quinlan's C5 algorithm groups the rules for each class together and exclude those that do not seem to contribute a great deal to the consistency of the set of rules as a whole. An end result is a minor number of rules that are easy to understand.

Taking Cost into Account

The error rate has been the primary metric in the debate to date to determine the fitness of rules and subtrees. However, in many implementations, the misclassification costs vary from class to class. Certainly, a false negative can be more dangerous in a medical test than a false positive; a terrifying Pap smear result that appears to have been a false positive at further testing is far preferable to undetected cancer. A cost function multiplies a weight indicating the cost of the misclassification by the likelihood of misclassification. Several tools allow such a cost function to be used in place of an error function to construct decision trees.

Further refinements to the Tree-Method decision

While they are not included in most commercial data mining software packages, there are some important refinements that are worth mentioning regarding the basic decision tree process.

Use More Than Only One Field at a Time

Many decision tree algorithms evaluate each split by a single variable. For several reasons, this method can be problematic, not least because it can lead to trees that have more nodes than required. Extra nodes are worrying because only the training records that enter a given node are available to trigger the subtree below it—the fewer examples of training per node, the less stable the resulting model. Suppose we're interested in a condition that is essential indicators for both age and gender. If the split root node is on age, then each child node contains just about half of the females. If the initial split is on gender, then each child node only contains about half of the old people. Several algorithms have been developed to allow the combination of multiple attributes to form the divider.

To minimize the complexity of the tree, one technique forms Boolean conjunctions of features. After discovering the feature that forms the best split, the algorithm searches for the feature that does the best job of enhancing the split when paired with the feature selected first. Apps continue to be added as long as the resulting break still indicates a statistically significant change. This may result in much more effective representation of the rules on classification. Consider, for example, the job of classifying the results of a vote according to whether the motion has been passed unanimously. Consider the situation where only three votes are counted, for simplicity. (The degree of simplification to be rendered increases only with the number of voters.) Table 6.1 includes all possible combinations of three votes and an additional column suggesting unanimity in the outcome.

Table 6.1 All Possible Combinations of Votes by Three Voters

FIRST VOTER	SECOND VOTER	THIRD VOTER	UNANIMOUS?
Nay	Nay	Nay	**TRUE**
Nay	Nay	Aye	**FALSE**
Nay	Aye	Nay	**FALSE**
Nay	Aye	Aye	**FALSE**
Aye	Nay	Nay	**FALSE**
Aye	Nay	Aye	**FALSE**
Aye	Aye	Nay	**FALSE**
Aye	Aye	Aye	**TRUE**

Figure 6.10 shows a tree that classifies the training data correctly, requiring five internal dividing nodes. Do not worry about the development of this tree, as this is irrelevant to the argument we are creating. In Figure 6.11, allowing the combination of features using the logical and function to form conjunctions yields the much simpler tree. The second tree shows another possible gain that may result from the use of field combinations. The tree now comes far closer to voicing the notion of unanimity that motivated the classes: "The decision is unanimous when all the electors are in agreement."

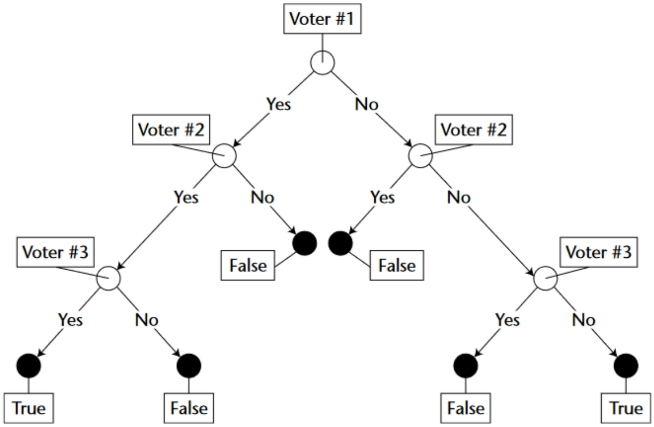

Voter #1
Yes
No
Voter #2
Voter #2
Yes
No
Yes
No
Voter #3
False
False
Voter #3
Yes
No
Yes
No
True
False
False
True

Figure 6.10 The best binary tree for the unanimity function when splitting on single fields.

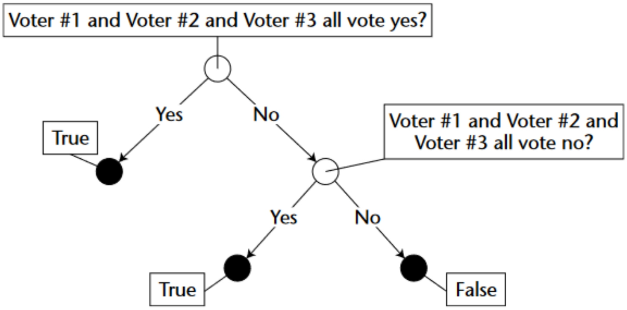

Voter #1 and Voter #2 and Voter #3 all vote yes?
Yes
No
True
Voter #1 and Voter #2 and Voter #3 all vote no?
Yes
No
True
False

Figure 6.11 Combining features simplifies the tree for defining unanimity.

A tree that can be understood at once is said to have a good "mental fit" by machine learning researchers. Some of the researchers in the field of machine learning attach considerable significance to this concept, but it appears that it is an artifact of the small, well-structured problems they are constructing their studies around. If a classification function is so easy in the real world that you can get your mind around the entire decision tree that describes it, you probably don't need to waste your time exploring it with powerful data mining software. We believe it is very important to be able to understand the rule that leads to any specific leaf; on the other hand, the ability to perceive a whole decision tree at a glance is neither necessary nor likely to be possible outside the laboratory.

Tilting the Hyperplane

Classification problems are also described in terms of geometry. This way of thinking is particularly common in cases where datasets have continuous variables for all fields. Any record in that definition is a point in a multi-dimensional space. Each field represents record location along one space axis. Decision trees are a way to carve the space into regions that are each labeled with a class. Any new record falling into one of the regions is thus classified. Traditional decision trees can only form rectangular regions that check the value of a single field at each node. A check of the form Y less than any constant in a two-dimensional space forms a region bounded by a line perpendicular to the Y-axis and parallel to the X-axis. Specific constant values allow the line to travel upwards and downwards, but the line stays horizontal. Similarly, a test on a single field in the space of higher dimensionality describes a hyperplane perpendicular to the axis defined by the field used in the test and parallel to all the other axes.

In a two-dimensional space, the resulting regions are rectangular, with only horizontal and vertical lines to deal with. The corresponding types are rectangular solids in three-dimensional space, and hyper-rectangles occur in any multi-dimensional space. The problem is that some objects do not fit perfectly into rectangular boxes. The problem is demonstrated in Figure 6.12: The two regions are really separated by a diagonal line; it takes a deep tree to produce enough rectangles to suit it properly. In this case, the true solution can easily be found by allowing the consideration of linear combinations of the attributes. Some software packages aim to tip the hyper-planes by basing their splits on a weighted sum of field values. For choosing the weights, there are a number of hill-climbing approaches.

Of course, regions that aren't easily captured even when diagonal lines are allowed are easy to come up with. Regions may have curved borders, and fields may need to be combined in more complicated ways (such as length by width multiplying to get area). There is no substitute for careful field selection to be inputs to the tree-building process and, where possible, the development of derived fields capturing known or suspected relationships by domain experts. Such derived fields that act as multiple other fields. Such derived fields inserted manually serve the same purpose as fields that automatically combine to tilt the hyperplane.

Neural Trees

One way of integrating information from several fields at each node is to provide a small neural network consisting of each node. For areas where rectangular regions do a bad job representing the classes' true shapes, neural trees may generate more accurate classifications while being faster than pure neural networks to train and score. From the user's point of view, this hybrid technique has more in common with variants of the neural-network than with variants of the decision-tree since, like other techniques of the neural network, it cannot justify its decisions. The tree still produces rules, but these are from the form F (w1x1, w2x2,w3x3, . . .) ≤ N, where F is the neural network's combination feature. These laws make neural network algorithms more sensitive than humans.

Piecewise Regression

Using Trees Another example of integrating trees with other modelling methods is a piecewise linear regression process in which each split is chosen in a decision tree to minimize the error of a simple regression model on the data at that node. For categorical target variables, the same approach can be extended to logistic regression.

6.3 Alternate Representations for Decision Trees

The traditional tree diagram depicts the internal structure of a decision tree is a very powerful way. Often other representations are more effective if the emphasis is more on the relative sizes and concentrations of the nodes. Box Diagrams While the tree diagram and comparison of Twenty Questions are helpful in visualizing some properties of decision-tree systems; a box diagram is more informative in certain situations. Figure 6.13 shows the box diagram representing a decision tree that tries to classify individuals as male or female based on their age and the films they've seen recently. The diagram may be seen as a kind of clustered array of two-dimensional scatter plots. At the root node of a decision tree, the first three-way split is based on which of three groups falls the most recently seen film by the survey respondent. The horizontal axis, in the diagram's outermost frame, represents that area. The outmost box is divided into parts at the next tree level, one for each node. The size or area of each segment is equal to the number of records falling into it. Next, each box's vertical axis is used to represent the field, which is used as the next splitter for that node. Overall, for each box, this will be a different area.

Now there's a new collection of boxes, each of which represents a node at the tree's third level. This cycle continues, separating boxes until each of the tree's leaves has its own box. Since decision trees often have non-uniform size, some boxes can more often be subdivided than others. Box diagrams make it easy to represent rules of classification which rely on a two-dimensional chart on any number of variables. The diagram of the lead is very descriptive. When we throw records onto the board, they fall into a common box and are categorized accordingly. A box chart allows us to look at different levels of detail on the results. Figure 6.13 indicates, at one glance, that there is a heavy concentration of males in the bottom left. Looking closer, we see some boxes that seem to be doing a very good job of classifying or storing a large number of documents. Viewed in this way, it is natural to think of the decision trees as a way to draw boxes around similar points categories.

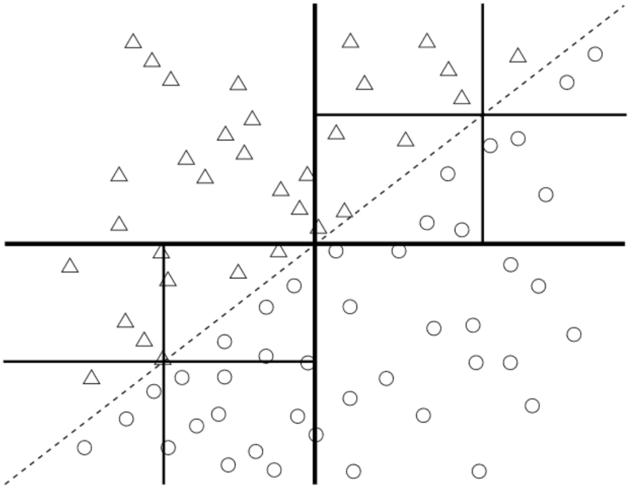


Figure 6.12 The upper-left and lower-right quadrants are easily classified, while the other two quadrants must be carved up into many small boxes to approximate the boundary between the regions.

All points inside a given box are marked the same way as they all comply with the law that determines the box. This contrasts with classical statistical classification methods such as linear, logistic, and quadratic discriminant, which attempt to divide data into groups by drawing a line or elliptical curve through the data space. This is a fundamental distinction: when there are many very different ways for a record to become part of the target class, statistical methods that use a single line to identify the boundary between classes are weak. Figure 6.14 illustrates this point using two dinosaur species.

The decision tree (represented as a box diagram) separated the stegosaurs from the triceratops with success. e.g., in the credit card industry, there are many methods of making consumers profitable. Some profitable customers have low transaction rates but, by default, hold high revolving balances. Others pay off their balance in full every month but are lucrative because of the large number of transactions they produce. But some do have few purchases, but often they make a major purchase and take several months to pay off. It can be equally beneficial for two very dissimilar clients. A decision tree will identify each separate group, mark them, and suggest the explanation for the profitability of each group by offering a summary of the box itself.

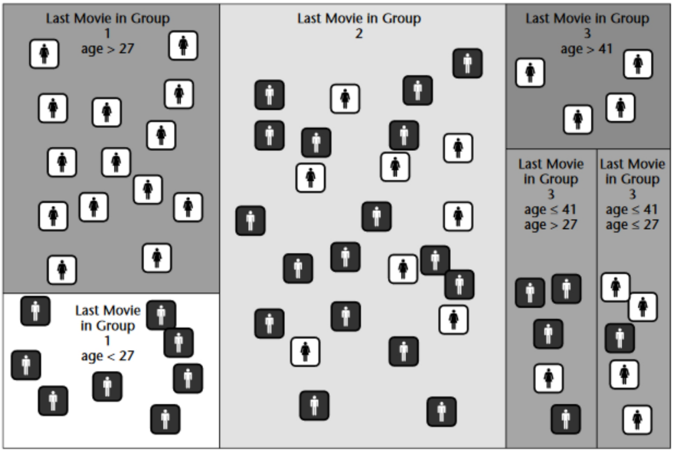

Last Movie in Group
1
age > 27
Last Movie in Group
2
Last Movie in Group
3
age > 41
Last Movie in Group
1
age < 27
Last Movie in Group
3
age ≤ 41
age > 27
Last Movie in Group
3
age ≤ 41
age ≤ 27

Figure 6.13 A box diagram represents a decision tree. Shading is proportional to the purity of the box; size is proportional to the number of records that land there.

Diagrams of the Tree Rings

The Enterprise Miner product from SAS Institute utilizes another clever representation of a decision tree. The diagram in Figure 6.15 looks like the tree is cut down, and we look at the stump.

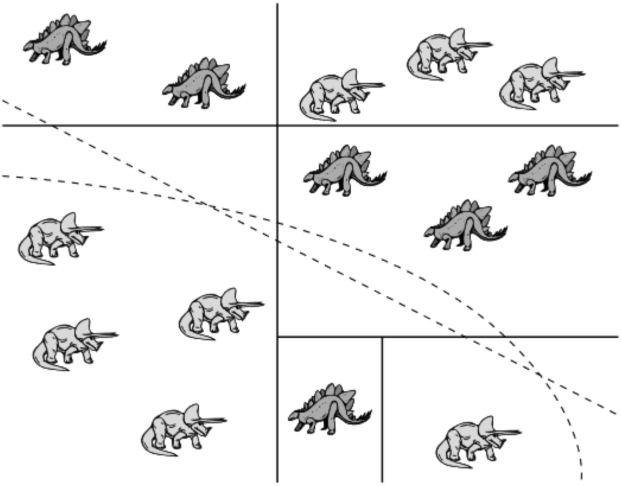


Figure 6.14 Often a simple line or curve cannot separate the regions and a decision tree does better.

The circle at the diagram's center represents the root node before any splits are done. Every focused ring moving out of the center represents a new level in the tree. The nearest ring to the middle represents a split root node. The length of the arc is equivalent to the number of records that take each of the two paths, and the shading reflects the purity of the node. The first break in the model that this diagram represents is relatively unbalanced.

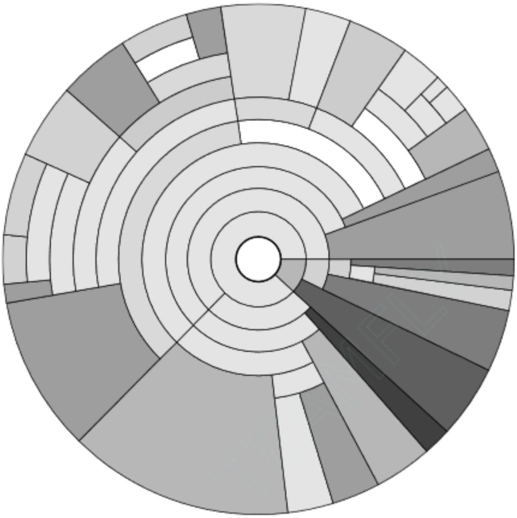


Figure 6.15 A tree ring diagram produced by SAS Enterprise Miner summarizes the different levels of the tree.

It divides the records into two classes, a large one with little difference in concentration from the parent population and a small one with high target class concentration. On the next point, this smaller node is split again, and one branch, represented by the thin, dark pie slice that stretches all the way to the outermost ring of the diagram, is a leaf node. The ring diagram shows the depth and complexity of the tree at a glance and shows the position of high concentrations on the target class. What it does not explicitly display are the rules that describe the nodes. When a user clicks on a specific portion of the diagram, the program shows those.

Chapter 7: Artificial Neural Networks of Data Mining

Artificial neural networks are common since many data mining and decision-support applications have a proven track record. Neural networks are a class of efficient, general-purpose tools readily applied to prediction, classification, and clustering — the "artificial" is typically removed. From forecasting time series in the financial world to diagnose medical problems, from identifying clusters of valued customers to detecting fraudulent credit card purchases, from reading numbers written on checks to forecasting engine failure rates, they have been applied across a wide variety of industries. Of course, the most efficient neural networks are the biological kind.

The human brain helps people to generalize from experience; computers, on the other hand, usually excel over and over in following specific instructions. Neural networks' benefit is that they fill this gap by modeling the neural interactions in human brains on a digital computer. Used in well-defined contexts, their ability to generalize and learn from data mimics our own capacity to learn from experience in some way. This skill is useful for data mining, and it also makes neural networks an exciting research field, promising new and better future results. But there is a downside. The outcomes of neural network training are internal weights that are spread across the network.

These weights provide no further insight into why the approach is true than describing our thought processes by dissecting a human brain. Maybe one day, sophisticated neural network sampling techniques will help to provide some clarity. Meanwhile, neural networks are better viewed as black boxes of internal workings that are as enigmatic as our brain's function. Like Oracle's responses at Delphi worshipped by the ancient Greeks, the responses generated by neural networks are often accurate.

They have a market value – a more critical aspect in many cases than giving an interpretation. This chapter starts with a little bit of history; neural network roots evolved from real attempts to model the human brain on computers. Then, before plunging into technical specifics, it describes an early case history of using this method for real estate evaluation. Most of the chapter introduces predictive modeling methods neural networks. Ultimately we see how they can also be used for undirected data mining. As always, a good place to start is at the top, with a bit of history.

History

In computer science, annals Neural networks have a fascinating past. The initial work on neuron functioning — biological neurons — had taken place in the 1930s and 1940s, before digital computers had even existed. In 1943, a neurophysiologist at Yale University, Warren McCulloch, and a logician, Walter Pitts, postulated a basic model to describe how biological neurons function and published it in a paper called "A Logical Calculus Immanent in Nervous Activity." While their focus was on explaining and understanding the anatomy of the brain, it turned out that this model influenced the field.

When digital computers were first available in the 1950s, computer scientists introduced models called perceptrons, based on McCulloch and Pitts' research. One example of a problem solved by these early networks was how to stabilize a broom that was standing upright on a moving cart by manipulating back and forth cart motions. As the broom begins to slip to the left, the cart learns to push to the left in order to hold it upright. While there have been some limited perceptron successes in the laboratory, the results have been disappointing as a general method for problem-solving.

One explanation of why early neural networks were of limited utility is that the most powerful computers of that period were less powerful than today's cheap desktop computers. Another explanation was that, as Seymour Papert and Marvin Minsky (two professors at the Massachusetts Institute of Technology) demonstrated in 1968, these simple networks had theoretical deficiencies. Because of these shortcomings, during the 1970s, the study of neural network implementations on computers significantly slowed down. Then in 1982, the California Institute of Technology's John Hopfield discovered backpropagation, a way to train neural networks that sidestepped the theoretical pitfalls of earlier approaches.

This growth has spurred a revival in research into neural networks. Work spread from laboratories into the business world through the 1980s, where it has since been applied to solve both operational problems — such as detecting fraudulent credit card purchases as they occur and identifying numerical amounts written on checks — and data mining challenges. Although artificial intelligence researchers were designing neural networks as a model of biological behavior, statisticians were taking advantage of computers to broaden the capabilities of statistical methods. For several forms of statistical analysis, a technique called logistic regression proved especially useful. Like linear regression, the logistic regression attempts to match the observed data to a curve.

However, it does use a function called the logistic function instead of a line. Logistic regression can be interpreted as special cases of neural networks, and even its more common counterpart linear regression. In addition, the whole neural network theory can be explained using statistical methods, such as distributions of probabilities, likelihoods, etc. However, this chapter tends more towards the biological model than towards statistical statistics for expository purposes. Neural networks became common in the 1980s, due to several factors converging.

Firstly, computing power was readily available, particularly where data was available in the business community. Second, by recognizing that they are closely connected to proven statistical approaches, analysts became more confident with neural networks. Thirdly, there were important data because operating systems were already automated in most businesses. Fourth, it was more important to practical applications than the holy grails of artificial intelligence. Creating tools to help people superseded the purpose of creating artificial individuals. Neural networks are and will continue to be, common data mining tools because of their proven usefulness.

Appraisal Of Real Estate

Neural networks have the potential to learn from the example in much the same way that experienced human experts do. The example below applies neural networks to solve a problem most readers are familiar with — real estate evaluation. Why should we want the appraisals automated? Clearly, automated appraisals may help real estate agents better fit prospective buyers to prospective properties, maximizing even novice agent productivity. Another usage would be to set up kiosks or web pages where prospective buyers would identify the homes they wanted — and get instant input on how much their dream homes cost. In the secondary mortgage market, maybe there is an unforeseen demand. Good, reliable evaluations are crucial to evaluating the risk of individual loans and loan portfolios, as one major factor that affects default is the proportion of the risk value of the property.

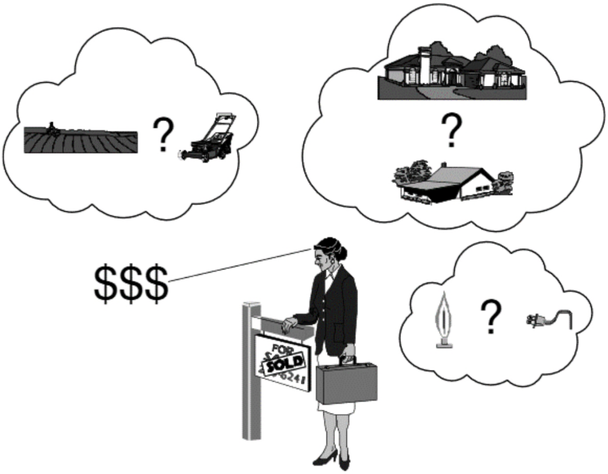
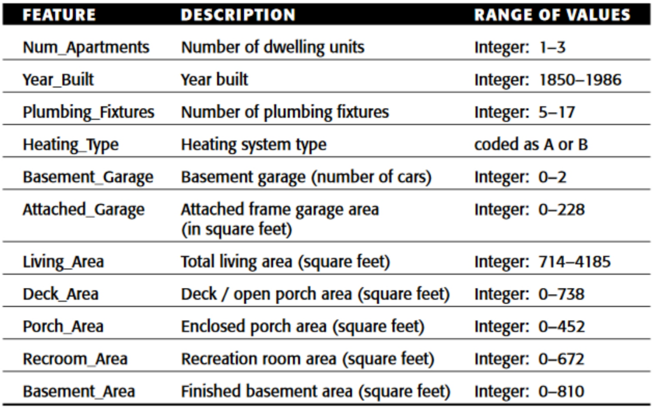


Figure 7.1 Real estate agents and appraisers combine the features of a house to come up with a valuation—an example of biological neural networks at work.

If the value of the loan reaches 100 percent of the market value, the default risk increases dramatically. How can the market value be measured after the loan has been made? To this end, the Federal Home Loan Mortgage Company, FreddieMac, has created a software called Loan Prospector that automatically performs such appraisals for homes throughout the United States. Initially, Loan Prospector was based on neural network technology developed by HNC, a San Diego company that has since merged into Fair Isaac. Back to the example.

This neural network imitates an appraiser who calculates a house's market value on the basis of property features (see figure 7.1). She knows houses are worth more than those in other areas in one part of the town. Other considerations in her mental estimation are additional bedrooms, a larger garage, the design of the home, and the size of the lot. She does not follow any fixed formula but combines her experience and knowledge of similar home selling rates. And, her understanding of the price of housing isn't static. She is aware of recent home sales prices around the area, and can identify market patterns over time — fine-tuning her estimate to suit the latest data.

Table 7.1 Common Features Describing a House



FEATURE	DESCRIPTION	RANGE OF VALUES
Num_Apartments	Number of dwelling units	Integer: 1–3
Year_Built	Year built	Integer: 1850–1986
Plumbing_Fixtures	Number of plumbing fixtures	Integer: 5–17
Heating_Type	Heating system type	coded as A or B
Basement_Garage	Basement garage (number of cars)	Integer: 0–2
Attached_Garage	Attached frame garage area (in square feet)	Integer: 0–228
Living_Area	Total living area (square feet)	Integer: 714–4185
Deck_Area	Deck / open porch area (square feet)	Integer: 0–738
Porch_Area	Enclosed porch area (square feet)	Integer: 0–452
Recroom_Area	Recreation room area (square feet)	Integer: 0–672
Basement_Area	Finished basement area (square feet)	Integer: 0–810

A good example of a human expert in a well-defined environment is the appraiser or real estate agent. Houses are defined by a fixed set of standard features that the expert takes into consideration and turns into an appraised value. In 1992, IBM researchers acknowledged this as a good neural network problem. Figure 7.2 exemplifies why.

A neural network takes specific inputs – in this case, the housing sheet information – and transforms them into a specific output, an appraised value for the property. Due to two reasons, the list of inputs is well defined: extensive use of the Multiple Listing Service (MLS) to exchange information about the housing market among different real estate agents, and standardization of housing specifications for mortgages sold in secondary markets. And the target production is well defined — a fixed sum of the dollar. In addition, there is a wealth of experience for teaching the network how to value a house in the form of previous sales.

The first and initial step in developing a neural network for calculating estimated housing prices is to evaluate a collection of features that influence the selling price. Table 7.1 lists some potential common features. Practically these features operate in a single geographic region for homes. To expand the assessment example to have homes in other neighborhoods, the input data will include zip code information, neighborhood demographics, and other quality-of-life measures for the community, such as school ratings and transport proximity. These additional features are not included here, to simplify the example.

Network training creates a model that can then be used to approximate the target value for unknown instances. Training provides the network with established examples (data from past transactions), so it can learn how to measure the selling price. The training examples include two additional features: home selling price and date of sale. It takes the selling price as the target variable. Using the date, the examples are divided into a preparation, validation, and test collection. An example from the training set is shown in Table 7.2.

The network training process is basically the weight adjustment process within it to arrive at the right combination of weights to make the predictions you want. The network begins with a random collection of weights, so it does very poorly in the initial context. However, the network is slowly doing a better and better job of approximating the target values in training set by reprocessing the training set over and over and changing the internal weights at each time to reduce the overall error. The network should cease training when the approximations no longer progress. This weight adjustment method is responsive to the representation of input data.

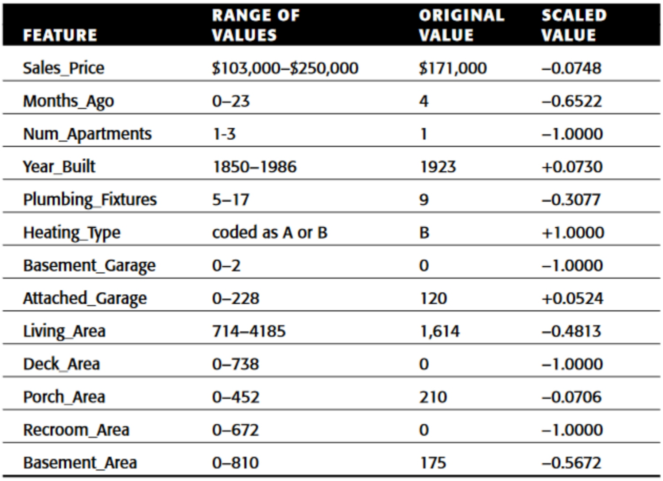
Consider, for example, a field in the data which measures the lot size. If lot size is expressed in acres, then the values may fairly be between about 1/8 and 1 acre. The same measurements would be 5.445 square feet to 43.560 square feet if measured in square feet. Neural networks, however, limit their inputs to small numbers for technical reasons, say between – 1 and 1.

For example, if an input variable assumes very large values compared to other inputs, then this variable dominates the target calculation. By reducing the weights on this input, the neural network wastes useful iterations to lessen its effect on the output. That is, the first "pattern" the network can find is that the variable in lot size has far greater values than other variables. As this is not especially important, using the lot size as measured in acres rather than square feet would be easier. This idea is widespread.

Typically small numbers will be inputs in the neural network. It is a good approach to restrict them to a certain limited range, such as – 1 to 1, which involves a mapping of all the values, both continuous and categorical, prior to network training. One way of representing continuous values is to make them fractions by subtracting the mean value of the range from the sum, dividing the result by the range size, and multiplying by 2. For example, in order to obtain a mapped value for Year Built (1923), subtract (1850 + 1986)/2 = 1918 (the mean value) from 1923 (the year the oldest house was built) and get 7.

Dividing by the number of years within the period (1986 – 1850 + 1 = 137) gives a scaled value, and then multiplying by 2 yields a value of 0.0730. This simple procedure can be extended to any continuous function to get a value of – 1 to 1. One way to map categorical features is for each group to be assigned fractions between – 1 and 1. In this data, the only categorical variable is Heating Type, so we can arbitrarily map B 1 and A to -1. If we had three values, we might assign 1 to – 1, another to 0, and the third to 1, although this solution has the downside that the three types of heating tend to have an order. Form – 1 shows up closer to form 0 than type 1. Next, Chapters addresses more ways of transforming categorical variables into numerical variables without inserting deceptive information. With these simple and easy techniques, it is possible to map all the fields for the previously displayed sample house record (see Table 7.2) and train the network.

Table 7.2 Sample Record from Training Set with Values Scaled to Range –1 to 1



FEATURE	RANGE OF VALUES	ORIGINAL VALUE	SCALED VALUE
Sales_Price	$103,000–$250,000	$171,000	–0.0748
Months_Ago	0–23	4	–0.6522
Num_Apartments	1-3	1	–1.0000
Year_Built	1850–1986	1923	+0.0730
Plumbing_Fixtures	5–17	9	–0.3077
Heating_Type	coded as A or B	B	+1.0000
Basement_Garage	0–2	0	–1.0000
Attached_Garage	0–228	120	+0.0524
Living_Area	714–4185	1,614	–0.4813
Deck_Area	0–738	0	–1.0000
Porch_Area	0–452	210	–0.0706
Recroom_Area	0–672	0	–1.0000
Basement_Area	0–810	175	–0.5672

Training is an iterating cycle across the training collection to change the weights. Often, each iteration is called one generation. If the network is trained, each generation's success must be calculated on the validation collection. Usually, on the validation set, earlier generations of the network perform better than the final network (that was optimized for the training set). This is due to overfitting and is a function of being so strong in neural networks. Neural networks are, in fact, one example of a universal approximator. That is, an appropriately complex neural network can approximate any function. Neural networks and decision trees can have this property that linear and logistic regression does not occur because they presume the underlying function in different shapes. As with other approaches to modeling, neural networks may learn patterns that only exist in the training set, resulting in overfitting. The training process remembers every set of weights calculated during each generation to find the best network for the unseen data. The final network comes from the generation which works best on the validation set, rather than the one which works best on the training set. If the performance of the model on the validation set is satisfactory, the configuration of the neural network is ready for use.

It learned from the training examples and worked out from all the inputs on how to determine the selling price. The model takes concise, suitably mapped information about a house and produces a production. There is one warning. The performance itself is a number between 0 and 1 (for a logistic activation function) or – 1 & 1 (for the hyperbolic tangent), which has to be reshuffled to the selling price range. For example, the value 0.75 could be multiplied by the range size ($147,000) and then added to the range base number ($103,000) to get an evaluation value of $213,250.

7.1 Neural Data Mining Networks

The example above shows the most common use of neural networks: constructing a model for classification or prediction. In this method, the steps are:

1. Identify the functions of the input and the output.
2. The inputs and outputs are converted in a limited range (– 1 to 1).
3. Build a network with a topology suitable for that.
4. Train the network on a representative set of examples for testing.
5. Use the validation set to select the weights set, which will minimize the error. 6. Using the test set to analyze the network, to see how well it performs.
6. Using a network-generated model to predict results for unknown inputs.

Fortunately, data mining software now automatically performs most of those moves. While it is not essential to have extensive knowledge of internal operations, there are some keys to effective network use.

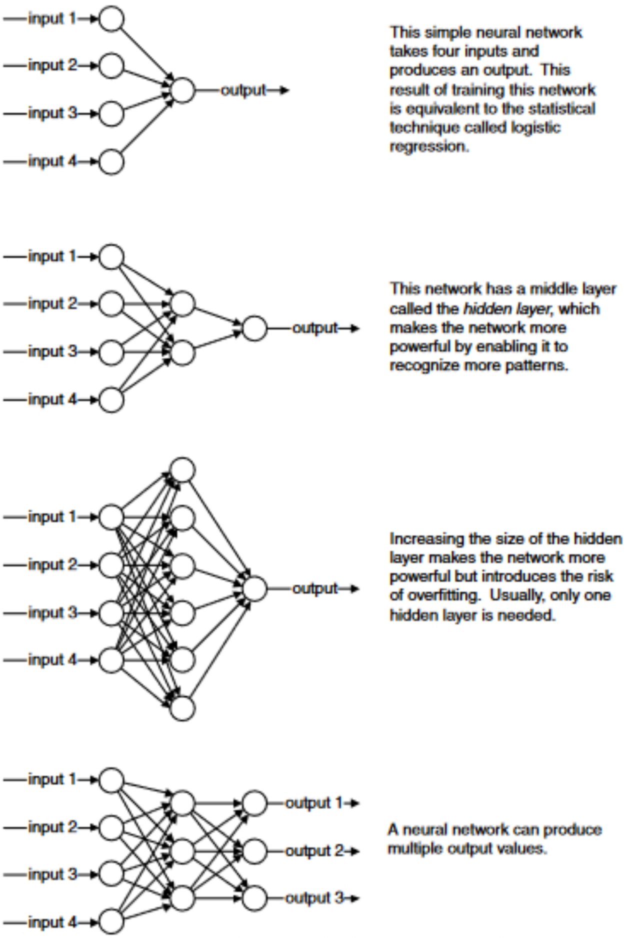
The most important problem, as with all predictive modeling devices, is selecting the right train-set. The second is representing the data in a way that maximizes the network's ability to identify patterns within it. The third is an analysis of network performance. Finally, learning some basic information about how they operate, such as topology of the network and training guiding parameters, will help make networks stronger. One of the dangers of any model used for prediction or classification is that as it gets old, the model becomes stale — and neural network models are no exception to this law.

For the example of the assessment, the neural network has learned about historical trends that allow it to predict the appraised value based on the content of the training set from descriptions of houses. There's no assurance that current market conditions suit those of last week, last month, or six months ago — when the training set may have been made. Every day new homes are purchased and sold, generating and adapting to market factors that are not present in the training package. A rise or the fall in interest rates, or an increase in inflation, may change the valuation values rapidly. Two factors complicate the question of keeping a neural network model up-to-date. Second, the model doesn't express itself readily in the form of laws, so when it has grown outdated, it might not be apparent.

The alternative is to insert more recent data into the neural network. One way is to return to training mode with the same neural network and start maintaining it new values. This is a valid and good approach if the network just needs to tweak outcomes like when the network is pretty close to being accurate, but you think you can boost its accuracy even further by giving it more recent examples. Another solution is to start over again by adding new examples into the training set (maybe deleting older examples) and training a whole new network, maybe even with a different topology (there is more discussion of network topologies later). This is acceptable because business dynamics may have fundamentally changed, and the trends found in the original training set no longer apply.

What Are Neural Nets?

Neural networks consist of simple units that imitate the actions of biological neurons found in nature in a simplified form, whether they constitute the brain of a human being or a frog. For example, it has been claimed that there is a unit within a frog's visual system that fires in response to fly-like motions and that there is another unit that fires in response to things around a fly's size. These two units are related to a neuron that fires when these two inputs have high combined value. This neuron is yet another input that activates tongue-flicking behavior. The basic concept is that each neural network has several inputs that the network integrates into a single output value, whether in a frog or a machine. Such units can be related to specialized nerves in the brains. Computers, however, are a bit simpler; as seen in Figure 7.3, the modules are actually connected together, and the outputs from some modules are used as inputs into others. All the examples in Figure 7.3 are examples of feed-forward neural networks, meaning that there is a one-way movement from the inputs to the outputs through the network, and there are no loops inside the network.



This simple neural network takes four inputs and produces an output. This result of training this network is equivalent to the statistical technique called logistic regression.

This network has a middle layer called the *hidden layer*, which makes the network more powerful by enabling it to recognize more patterns.

Increasing the size of the hidden layer makes the network more powerful but introduces the risk of overfitting. Usually, only one hidden layer is needed.

A neural network can produce multiple output values.

Figure 7.3 Feed-forward neural networks take inputs on one end and transform them into outputs.

Feed-forward networks are the simplest and most versatile form of guided modeling network. There are three specific questions you can think about it:

- What are units, and how are they doing? That is, what activation function does it have?
- How are the units connected? That is, what is network topology?
- How does the network learn pattern recognition?

That is, what is propagation back and, more precisely, how is the network trained? The answers to these queries provide the context for understanding basic neural networks, an understanding that provides guidance from this powerful data mining technique to get the best results.

What is a Neural Networking Unit?

Figure 7.4 demonstrates the major artificial neuron characteristics. The unit combines its inputs into a single value and converts them to generate the output; these are called the activation function together. The most popular activation functions are based on the biological model, where output remains very small until a threshold state is reached by the combined inputs. When the combined inputs exceed the threshold, the device is switched on and high performance. Like its biological equivalent, the unit in a neural network has the property that small changes in the inputs can have fairly large effects on output when the combined values are within some middle range. In comparison, significant input changes may have little effect on performance, when the combined inputs are far from the middle range. This property, in which minor changes sometimes matter and sometimes they don't, is an example of nonlinear behavior.

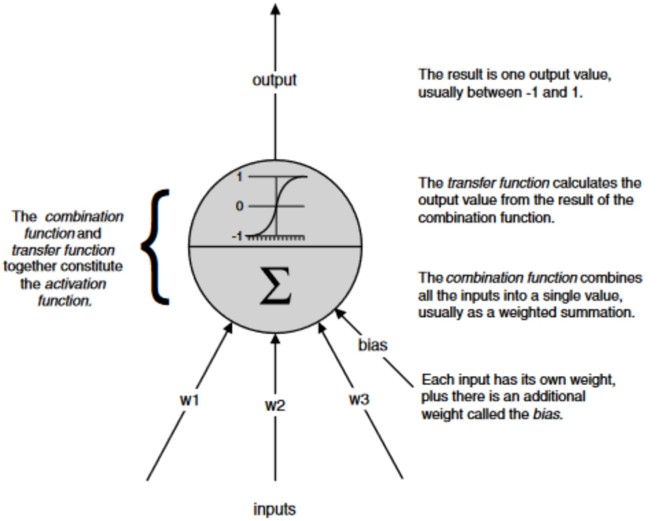

output
The result is one output value, usually between -1 and 1.
1
0
-1
The transfer function calculates the output value from the result of the combination function.
The combination function and transfer function together constitute the activation function.
Σ
The combination function combines all the inputs into a single value, usually as a weighted summation.
bias
w1
w2
w3
Each input has its own weight, plus there is an additional weight called the bias.
inputs

Figure 7.4 The unit of an artificial neural network is modeled on the biological neuron. The output of the unit is a nonlinear combination of its inputs.

Neural networks' strength and complexity derive from their nonlinear nature, which in turn derives from the particular activation mechanism used by the constituent neural units. There are two sections to the Activation function. The first component is the task of integrating all the inputs into a single value. -- input into the unit has its own weight, as shown in Figure 7.4. The weighted sum is the most common combination method, in which each input is multiplied by its weight, and these products are added together.

Many mixture functions are also useful and include the weighted input maximum, the minimum, and the value logical AND or OR. While there is a lot of versatility in selecting combination functions, in many cases, the standard weighted sum works fine. This option dimension is a common feature of neural networks. Their basic structure is very versatile, but in practice, the defaults that suit the original biological models, such as the weighted sum for combination function, work well.

The second component of the activation function is the transfer function, which gets its name from the fact that the value of the combination function is passed to the unit output. Figure 7.5 contrasts the three traditional tangent transfer functions: sigmoid (logistic), linear, and hyperbolic.

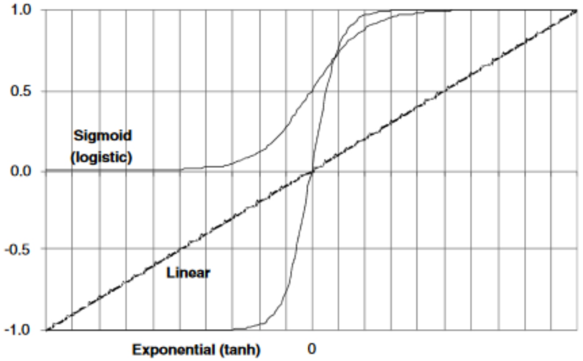


Figure 7.5 Three common transfer functions are the sigmoid, linear, and hyperbolic tangent functions.

The particular values assumed by the transfer function are not as important as the general functional type. From our viewpoint, the least important is the linear transfer function. A feed-forward neural network consisting only of units with linear transfer functions and a weighted sum combination function does a linear regression, indeed. Sigmoid functions are S-shaped functions, of which the logistic and hyperbolic tangent are the two most common ones for neural networks. The main difference between them is the range of their outputs, for the logistic between 0 and 1 and for the hyperbolic tangent between -1 and 1—the functions of logistic and hyperbolic tangent transfer work in a similar manner.

Although not linear, their conduct appeals to the statisticians. If the complete weighted sum of all inputs is equal to 0, then such functions are equivalent to a linear function. Statisticians love linear systems, and they enjoy near-linear systems just as well. As the magnitude of the weighted sum is greater, these transfer functions saturate slowly (in the case of logistics, to 0 and 1; in the case of the hyperbolic tangent, to – 1 and 1).

This conduct corresponds to a progressive transition from a linear input model to a nonlinear model. In short, neural networks have the potential to do effective modeling work on three types of problems: linear issues, near-linear issues, and nonlinear issues. There is also a relationship between the activation function and the range of input values, as discussed in the sidebar "Sigmoid

Functions and Ranges for Input Values." A network that contains units with different transfer functions, a topic that we will return to later when discussing the topology of the network. Often sophisticated tools allow experimenting with certain functions of combination and transfer. Many functions have behavior that is substantially different from standard units. Playing with various types of activation functions can be interesting and even helpful. However, if you don't want to bother, you can trust the standard functions that have been effective for many applications on the neural network.

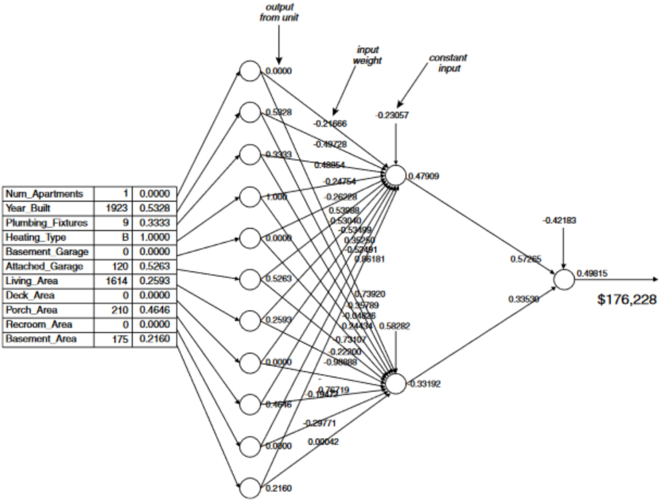
Feed-Forward Neural Networks

A feed-forward neural network calculates input values, as shown in Figure 7.6. This network's topology, or structure, is typical of networks that are used for prediction and classification. The units are divided into three strata. The layer on the left is attached to the inputs, and the input layer is labeled. -- unit in the input layer is connected to exactly one source field, usually mapped within the range – 1 to 1. The input layer doesn’t actually do any task at all in this case. Every unit on the input layer copies its input value to its output. If that is the situation, then why are we even trying to discuss it here? It is an essential part of the Neural Network vocabulary. The input layer reflects, in practical terms, the process of mapping values into a rational range. It's worth including them for that purpose alone since they are a reminder of a very significant aspect of the effective use of neural networks.

The next layer is called the secret layer since it is neither related to the inputs nor to the network output. Usually, every unit in the hidden layer is completely connected to all the units in the input layer. Because this network contains standard units, the units in the hidden layer determine their output by multiplying by their corresponding weight the value of each input, adding these up, and applying the transfer function.

A neural network can have numbers of hidden layers, but one hidden layer is sufficient in general. The broader the layer (i.e., the more units it includes), the greater the network's ability to detect patterns. However, this greater ability has a downside, because in the training cases, the neural network can memorize patterns-of-one. We want the network to generalize and not memorize on the training set. The secret layer should not be too broad for this to be done.

Note that each of the units in Figure 7.6 has an additional input coming down from above. This is the constant data, often referred to as bias, and always set to 1. Like other inputs, it has a weight and is included in the work of the combination. The bias acts as a global offset, which helps the network understand trends better. The training process changes the weights at constant inputs, just as it does on the network's other weights.



Num_Apartments	1	0.0000
Year_Built	1923	0.5328
Plumbing_Fixtures	9	0.3333
Heating_Type	B	1.0000
Basement_Garage	0	0.0000
Attached_Garage	120	0.5263
Living_Area	1614	0.2593
Deck_Area	0	0.0000
Porch_Area	210	0.4646
Recroom_Area	0	0.0000
Basement_Area	175	0.2160

output from unit
input weight
constant input
0.0000
0.5328
0.3333
1.000
0.0000
0.5263
0.2593
0.0000
0.4646
0.0000
0.2160
-0.21666
-0.49728
0.48854
-0.24754
-0.26228
0.53988
0.53040
-0.53499
0.35250
-0.52491
0.86181
-0.23057
0.47909
0.73920
-0.35789
-0.04826
-0.24434
0.58282
-0.73107
-0.22200
-0.98888
-0.76719
-0.19472
-0.29771
0.00042
-0.33192
-0.42183
0.57265
0.33530
0.49815
$176,228

Figure 7.6 The real estate training example shown here provides the input into a feed-forward neural network and illustrates that a network is filled with seemingly meaningless weights.

Note that each of the units in Figure 7.6 has an additional input coming down from above. This is the constant data, often referred to as bias, and always set to 1. Like other inputs, it has a weight and is included in the work of the combination. The bias acts as a global offset, which helps the network understand trends better. The training process changes the weights at constant inputs, just as it does on the network's other weights. The final unit on the right is the output layer since it is connected to the neural network output. It's completely connected to the secret layer of all the devices. Most of the time, a single value is determined using the neural network, and there's just one unit in the output layer and the value. To understand the output, we need to map the value back.

In Figure 7.6, we need to translate 0.49815 back to a value between $103,000 and $250,000 for the network. That is equal to $176,228 and is very similar to the real value of $171,000. The output layer uses a simple linear transfer method in some implementations, so the output is a weighted linear combination of inputs. That removes the need to map outputs. It is likely possible to have more than one device at the output layer. A department store chain, for example, needs to forecast the probability that consumers will buy items from different departments, such as women's wear, decor, and entertainment. The stores want this information to be used to schedule promotions and to guide target mailings.

They could set up the neural network shown in Figure 7.7 to make that prediction. This network has three outputs, each department having one. The outputs are a tendency for the customer to make his or her next purchase from the associated department as defined in the inputs.

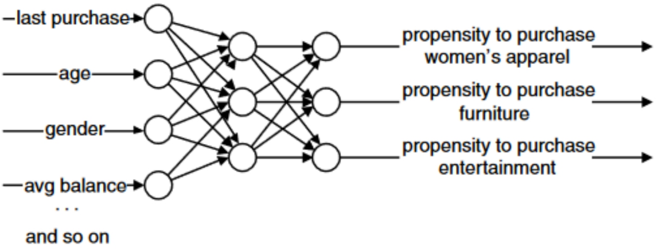

last purchase
age
gender
avg balance
. . .
and so on
propensity to purchase women's apparel
propensity to purchase furniture
propensity to purchase entertainment

Figure 7.7 This network has with more than one output and is used to predict the department where department store customers will make their next purchase.

The network measures three values after feeding the inputs into the network for a client. How will the department store deter the best deal or deals to sell the customer in light of all these outputs? Some specific ways of working with multiple model outputs are:

Take the department with the maximum value corresponding to the production. Take divisions with the top three values corresponding to the outputs.

- All departments are corresponding to outputs above a certain threshold value.
- Take all divisions are corresponding to units, which are a certain percentage of the overall value unit.
- All those possibilities fit well and, in various cases, have their strengths and disadvantages.

There's no one answer correct that always works. In practice, on the test range, you want to try out some of these possibilities to decide which works best in a given scenario. The topology of feed-forward neural networks differs further. The input layers are also connected directly to the output layer. In this case, there are two components to the network. Such direct connections serve as standard regression (linear or logistic, depending on the output layer activation function). The construction of more common statistical models is useful here. Then the secret layer serves as an approximation to the mathematical model.

How does a neural network use backpropagation to learn?

Training a neural network is the process of setting the best weights on the edges which connect all the network units. The purpose is to use the training set to measure weights, where the network output is as similar or related as possible to the desired output for as many of the examples as possible in the training set. Although backpropagation is no more the preferred method of weight change, it offers insight into how training works or operates, and it was the original method of training feed-forward networks. The following three phases are at the center of backpropagation:

1. The network gets an example of the training and measures the output or outputs using the current weights in the network.

2. Then the backpropagation calculates the error by taking the difference between the result measured and the predicted result (actual result).

3. The error is fed back via the network, and the weights are changed to mitigate the error — hence the transmission of the name

back as the errors are returned from the network.

The backpropagation algorithm tests the network's overall error by comparing the values produced on each example of training to the actual value. It then changes the output layer's weights to reduce the error, but not eliminate. The algorithm hasn't finished, though. It then assigns the blame to the network's earlier nodes and changes the weights that link those nodes, thereby further minimizing overall error. There's no significant precise process for assigning responsibility. Suffice it to say that backpropagation uses a complicated mathematical procedure which requires the activation function to take partial derivatives.

How does a machine change its weights in light of the error? It calculates whether the weight changes at each input will increase the error or decrease it. Then, the unit changes each weight to reduce the error but not remove it. The changes in the training set for each example gradually nudge the weights towards their optimum values. Note, the goal is to generalize and recognize input patterns, not memorize the training set. It is like a leisurely stroll instead of a mad-dash scramble to change the weights. After providing enough training examples over enough generations, the weights on the network are no longer dramatically increasing, and the error is no longer decreasing. This is the point where the training stops; the network has mastered input pattern recognition.

This weight adjustment technique is termed the generalized delta law. The use of the generalized delta rule requires two essential parameters. The first is momentum, referring to the tendency of the weights within each unit to shift the "direction" in which they move. That is, every weight recalls whether it's getting bigger or smaller, and the momentum is trying to keep it going in the likewise direction. A high momentum network is responding slowly to new examples of training that want to reverse the weights. If the momentum is weak, then weights will oscillate freer.

The learning rate regulates the speed at which weights change. The best approach to learning rate is to get started big and slowly decrease as the network is being educated. The weights are random at first, so large oscillations are useful to get the best weights in the vicinity. However, as the network reaches the optimum solution, the learning rate will decrease, so that the network can fine-tune to the right weights. Researchers discovered hundreds of variants for neural networks learning Growing of these methods has its pros and cons.

They are searching in all cases for a technique that trains networks rapidly to find an optimal solution. Some neural network packages provide several training methods, enabling users to experiment on their problems with the best solution. One of the risks is slipping into something called a local maximum for some of the training techniques. This happens when the network produces okay results for the training set, and weight modification no longer enhances network efficiency. There is, however, some other weight combination — significantly different from those in the network — that yields a much better solution. This is similar to attempting to climb to the top of a mountain by taking the steepest route at each turn and realizing you have climbed just to the peak of a small hill. There is a conflict between finding the best local solution and finding the best global solution. Learning rate and momentum management help to find the right solution.

Heuristics Feed-forward, Backpropagation Networks

Also, with advanced neural network packages, it takes some effort to get the best output from a neural network. This segment covers several heuristics for building a network to achieve good results. The number of units in the secret layer probably is the biggest decision. The more units the network will understand, the greater the patterns. That would argue for a hidden layer very big. There's one downside, though. The network might end up memorizing the training set, rather than generalizing from it. Things aren't better in this situation. Luckily when a network becomes overtrained, you can detect it. If the network performs on the training set very well but does much worse on the validation set, then this is an indicator that the training set has been memorized. How high is the secret layer? The true answer is nobody knows. It depends on the data, patterns to be identified, and network type.

Since overfitting is a major problem for customer data networks, we usually don't use secret layers greater than the number of inputs. Experimenting with one, two, and three nodes in the hidden layer is a good starting point for many problems. This is feasible, especially as training neural networks now takes seconds or minutes, rather than hours. If adding more nodes increases network efficiency, then it could be best to add larger nodes. Lower the layer size when the network is overtraining. If this isn't accurate enough, increase its scale. But beginning with one hidden node for each class may be useful when using a classification network.

The scale of the training set is a further consideration. To include the input ranges for each function, the training set must be sufficiently broad. Moreover, for every weight in the network, you want to have many training examples. There are h * (s + 1) + h + 1 weights in the network for a network with s input units, hidden units and 1 output (each hidden layer node has a weight for each input layer connection, an additional weight for the bias, and then a relation to the output layer and its bias). For example, if the hidden network contains 15 input features and 10 units, then there are 171 weights in the network. For each weight, there should be at least 30 examples but a better minimum is 100. The training set will have to have at least 17,100 rows for this example. Finally, the learning rate and momentum parameters are very necessary to get good results out of a network using the training algorithm for backpropagation (it's easier to use conjugate gradient or similar approach). To make major changes to the weights, the learning should initially be set high. As the training continues, the learning rate will fall to fine-tune the network. The momentum parameter helps the network to push faster towards a solution, preventing oscillation across weights, which are less useful.

Selecting the Training Set

The training set consists of records already established for their predictive or classification values. For all data mining modeling, the selection of a good training set is important. A weak training set is dooming the network, irrespective of any other research that enters its development. Luckily in choosing a correct one, there are just a few things to consider.

Values for All Features

The most significant of these considerations is that the training set has to cover the full spectrum of values for all features the network might encounter, including the performance. In the case of property valuation, this means that cheap houses and costly houses, large houses and small houses, and houses with and without garages are included. It is a good method to provide several examples in training set for each value of a categorical feature and for values across the ranges of discrete and continuous functions ordered.

This is valid regardless of whether the functions are currently used as network inputs. For example, lot size could not be selected as a variable input in the network. The training set will still have examples from all sorts of lot sizes, however. Possibly a network trained on smaller lot sizes (some of which may be low priced and some high priced) would not do a good job on mansions.

Amount of Feature

The number of features of the inputs influences neural networks in two ways. Firstly, the more features that are used as inputs into the network, the bigger the network needs to be, the greater the chance of overfitting, and the size of the training set. Second, the more features it takes to converge to a set of weights, the longer the network is. And the weights are less likely to be ideal, with too many functions. This problem of selecting variables is a common problem for statisticians. We consider in practice that decision trees (discussed in Chapter 6) provide a reasonable method of selecting the best variables. A pleasant function of SAS Enterprise Miner is shown in Figure 7.8. By linking a neural network node to a node of a decision tree, the neural network uses only the variables selected by the decision tree. An alternative solution is an intuition. Start off with a handful of sensible variables—experiment with trying to see which of the other variables boost the model.

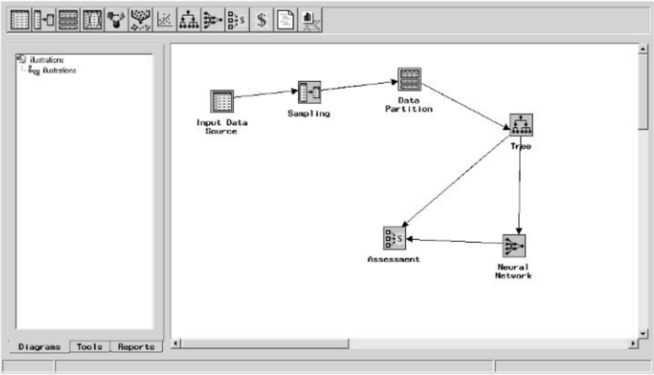


Figure 7.8 SAS Enterprise Miner provides a simple mechanism for choosing variables for a neural network—just connect a neural network node to a decision tree node.

In several cases, calculating new variables that represent particular aspects of the business problem is useful. For example, in the real estate example, we might deduct the house's square footage from the lot size to get an idea of how big the yard is.

Training Set Size

The more features the network has, the more training examples it takes to get good coverage of trends in the data. Unfortunately, there is no clear rule for communicating a relationship between the number of traits and the size of the training set. However, usually, a minimum of a few hundred examples are required to help each feature with sufficient coverage; it is not uncommon to provide several thousand. The authors have experimented with neural networks with just six or seven inputs, but with hundreds of thousands of rows in their training collection. When the training set isn't large enough, neural networks appear to over-adapt the data. Overfitting is likely to happen when there are fewer exercise instances than weights in the network. This poses a problem because on the training set, the network will perform very, very well, but on unknown data, it will fail spectacularly. Of course, the drawback to a sufficiently broad training set is that it takes longer to train the neural network. You that get better models in a given amount of time by using fewer input features and a smaller training set, and playing with different combinations of features and network topologies rather than using the largest training set possible that leaves no room for experimentation.

Output Number s

In most training scenarios, there are usually far more inputs coming in than outputs going out, so good input coverage results in good output coverage. It is also very necessary that there are several examples from the network for all possible performance values. On top of that, the number of training examples should be around the same for each potential performance. This can be crucial when determining which training set to use. For example, if the neural network is to be used to detect uncommon but significant events — failure levels in a diesel engine, fraudulent use of a credit card, or who will respond to a credit line offer for home equity — then the training set must provide enough examples of these rare events. A random sample of available data might not be enough, as typical examples would swamp the rare examples. The training set needs to be balanced by oversampling the unusual cases to get through this. A training set consisting of 10,000 "good" examples and 10,000 "poor" examples provides better results for this form of a problem than a randomly chosen training set of 100,000 good examples and 1,000 bad ones. Using the random sample of the training set, after all, the neural network will probably assign "ok" regardless of the input — and be 99 percent of the time right.

7.2 Neural Network For Time Series

Naturally, in many business concerns, the data falls into a time sequence. Examples of these series are the closing price of IBM stock, the regular value of the Swiss franc to the exchange rate of the US dollar, or a prediction of the number of customers who will be involved in the future at any given date. Anyone who is able to predict the next value for financial time series, or just whether the series heads up or down, has a huge advantage over other investors. While prevalent in the financial sector, in other fields, such as forecasting and process management , time series exist. However, financial time series are the most studied since a marginal advantage in predictive ability translates into significant income.

Neural networks are easily adapted for analysis of the time series, as shown in Figure 7.12. The network is trained or prepared on data from the time-series, starting at the data's oldest level. The training set then moves to the second oldest point and the oldest point goes to the input layer's next set of units, and so on. The network trains as a feed-forward, backpropagation network attempting at every stage to predict the next value in the sequence.

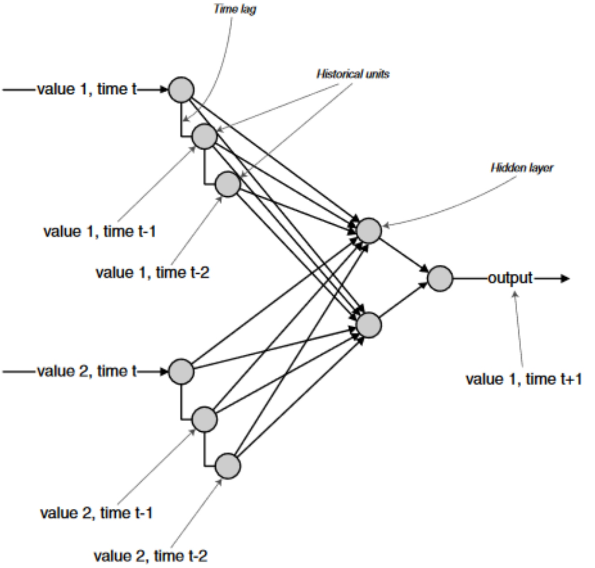

Time lag
value 1, time t
Historical units
Hidden layer
value 1, time t-1
value 1, time t-2
output
value 2, time t
value 1, time t+1
value 2, time t-1
value 2, time t-2

Figure 7.12 A time-delay neural network remembers the previous few training examples and uses them as input into the network. The network then works like a feed-forward, back propagation network.

Notice that the network time series isn't limited to data from a single time series. It can require multiple inputs. For example, to estimate the Swiss franc's value to the U.S. dollar exchange rate, other time-series details may be used, such as the number of the previous day's transactions, the U.S. dollar to the Japanese yen exchange rate, the stock exchange closing value, and the day of the week. Furthermore, non-time-series data, such as the estimated rate of inflation in the countries during the period under investigation, may also be candidate features.

Historical unit numbers govern the length of patterns that the network may remember. For example, having 10 historical units on a network predicting a favorite stock's closing price would allow the network to recover trends that occur within 2-week timeframes (because exchange rates are set only on weekdays). It might not be a good approach to rely on such a network to forecast the value for 3 months in the future and is not recommended. Actually, a feed-forward network can be made to work as a time-delay neural network, by modifying the input. Consider the 10 day history time sequence, as shown in Table 7.5.

Table 7.5 Time Series

DATA ELEMENT	DAY-OF-WEEK	CLOSING PRICE
1	1	$40.25
2	2	$41.00
3	3	$39.25
4	4	$39.75
5	5	$40.50
6	1	$40.50
7	2	$40.75
8	3	$41.25
9	4	$42.00
10	5	$41.50

Table 7.6 Time Series with Time Lag

DATA ELEMENT	DAY-OF-WEEK	CLOSING PRICE	PREVIOUS CLOSING PRICE	PREVIOUS-1 CLOSING PRICE
1	1	$40.25		
2	2	$41.00	$40.25	
3	3	$39.25	$41.00	$40.25
4	4	$39.75	$39.25	$41.00
5	5	$40.50	$39.75	$39.25
6	1	$40.50	$40.50	$39.75
7	2	$40.75	$40.50	$40.50
8	3	$41.25	$40.75	$40.50
9	4	$42.00	$41.25	$40.75
10	5	$41.50	$42.00	$41.25

There are two aspects to the network: the day of the week and the closing date. Creating a time series with a time lag of three includes applying new features to the historical values lagged behind. (day-of-week doesn't have to be copied, because it doesn't really change.)

The result is in Table 7.6. Now, these data can be inserted into a feed-forward, backpropagation network without any special time series support.

How to Know What Goes Inside a Neural Network?

Neural networks are very invisible. Even understanding all the weights around the network on all the nodes doesn't provide any insight into why the network is generating the results it produces. There is some metaphysical appeal to this lack of understanding — after all, we don't understand how human consciousness emerges from our brain neurons. However, as a practical matter, obscurity impairs our ability to interpret the effects a network produces. If only we could suggest it to tell us how it takes the type of rules to make its decision. Unfortunately, the same nonlinear features that make neural network nodes so efficient also render them unable to follow simple laws. Research into the extraction of rules from networks can eventually yield unmistakably good results. The qualified network itself has been the law until then, and other approaches are required to peer inside to understand what's going on. We may use a technique called sensitivity analysis to get an understanding of how opaque models operate. Analysis of sensitivity does not have clear rules, but it does show the relative importance of inputs to the network effect. Analysis of sensitivity uses the test set to assess how sensitive the network output is to each input. The underlying measures are:

1. Find the mean value for any input. We can think of that average value as the test set hub.

2. Measure network output when all input is at its average value.

3. Measure network output by changing each input, one at a time, to be at its minimum and maximum values (usually – 1 and 1, respectively).

For some inputs, the network output changes very little for the three values (minimum, average, and maximum). These inputs are not accessible to the network (at least where all other inputs are at their average value). Other inputs have a significant impact on network performance. These inputs are open to the Network. The sum of output adjustment tests the network's sensitivity for each input. Using such metrics for all the inputs provides a quantitative measure of each feature's value. Naturally, this approach is fully observational and looks only independently in each aspect. Neural networks are of special interest precisely because they can take account of interactions between variables.

Variations exist about this method. Eventually, it is possible to change the values of two or three functions to see which combinations of features are of particular interest—at times, starting from a position other than the test set's core is useful. For example, the study could be replicated for function minimal and maximum values to see how responsive the network is at the extremes. If the study of sensitivity shows substantially different results for these three cases, there are higher-order effects in the network that take advantage of feature combinations. When using a feed-forward, backpropagation network, sensitivity analysis can take advantage of the error measurements measured during the learning process, rather than having to individually evaluate each function.

The validation set is filled into the network to generate the output, and to calculate the error; the output is compared to the expected output. The network signals then propagates the error back through the units to keep track of the sensitivity of input, not to change any weights. The error is a sensitivity proxy which determines how much each input affects network output. The aggregation of these sensitivities in the entire test set dictates the inputs will have a greater impact on the output. However, in our experience, the values generated in this way are not especially useful for network understanding.

Self-Organizing Maps

Self-organizing maps (SOMs) are a set of neural network that used for undirected tasks in data mining, such as cluster detection. Dr. Tuevo Kohonen, a Finnish scholar, invented self-organizing maps, also called Kohonen Networks. These networks also identify clusters in data while they were initially used for images and sounds. They are based on the identical underlying units as feed-forward, backpropagation networks, but in two respects, SOMs are very different. They have a different topology, and the method of learning behind propagation is no longer valid. They've got a totally different way of teaching.

What is a Map of Self-organization?

The self-organizing map (SOM), an example shown in Figure 7.13, is a neural network that can identify patterns that are unknown in the data. The simple SOM has an input layer and an output layer, much like the networks that we've already looked at. -- unit in the input layer is connected to a single source, just as the predictive modeling networks do. Each unit in the SOM, like those networks, also has an independent weight associated with each incoming link (this is actually a property of all neural networks). The similarity between SOMs and feed-forward, backpropagation networks does end here, however.

The output layer is made up of several components, rather than only a few. In the output layer, each unit is connected to all the units in the input layer. In a grid, the output layer is arranged as if the units were on a checkerboard in the squares. Although the units in this layer are not related to each other, the grid-like structure plays an important role in the SOM's training, as we'll see soon. How does a SOM accept patterns? Imagine one of the carnival booths where you'll throw balls at a wall riddled with holes. If you land the ball in one of the holes, then you've got your pick of prizes. Training a SOM is like being blindfolded at the booth, and at first, there are no holes in the wall, very similar to the situation when you start searching for patterns in vast quantities of data and don't know where to start. Each time you throw the ball, a little bit of it dents the wall. When enough balls land in the same area, the indentation gradually breaks through the wall, creating a hole.

Now, it's going through the hole when another ball lands at that spot. You get a prize — this is a cheap stuffed toy at the carnival, with a SOM; it's a recognizable band. Figure 7.14 shows how a simple SOM works on this. When a member of training set is introduced to the network, the values flow to the units in the output layer through the network. The units in the output layer that compete with each other, and the one with the highest "wins" value. The incentive is to change the weights leading up to the winning unit to reinforce the input pattern response. That's like making a tiny dent in the network.

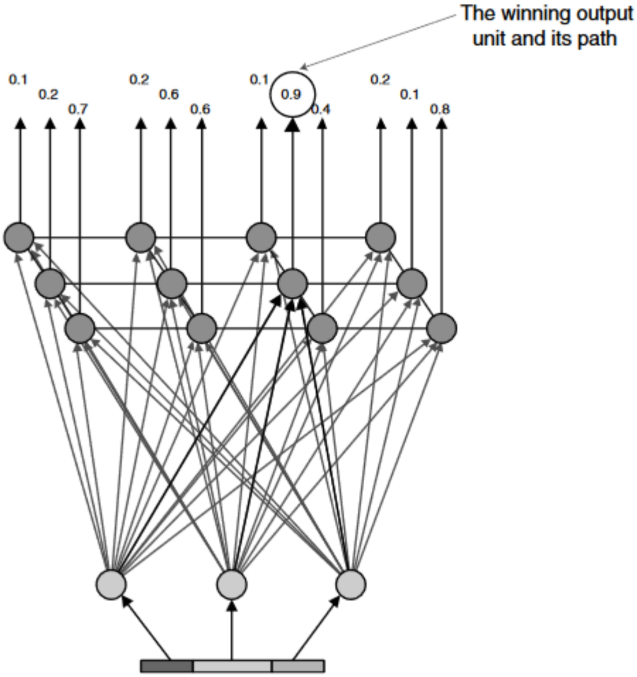

The winning output unit and its path

Figure 7.14 An SOM finds the output unit that does the best job of recognizing a particular input.

There is one more aspect of network preparation. The weights for the winning unit are not only adjusted, but the unit weights in its immediate neighborhood are also adjusted to improve their response to the inputs. This adjustment is regulated by a neighborliness parameter that controls neighborhood size and adjustment amount. The neighborhood is very big at first, and the changes are fantastic. The neighborhoods and changes are declining in size as the training continues. Indeed, neighborliness has many practical consequences. One is that the output layer is more like a woven fabric, although the units are not attached directly to each other. Clusters identical to each other should be smaller than clusters, which are more dissimilar. More significantly, however, neighborliness allows one unit group to represent a single cluster. The network will try to find as many clusters in the data as there are units in the output layer without this neighborliness — introducing bias into the cluster detection.

A SOM usually recognizes fewer clusters than the units it has produced. This is inefficient when the network is used to allocate new records to the clusters, as the new inputs are fed into unused units in the output layer via the network. We apply the SOM to the validation collection to decide which units are currently in use. The validation set members are fed in via the network, in each case keeping track of the winning unit. Units without hits are destroyed or with very few hits.

Eliminating those units improves the network's runtime performance by reducing the number of calculations needed for new instances. When the final network is in place — the output layer being limited only to the units representing unique clusters — it can be extended to new instances. An unknown instance is fed into the network, and at the output unit with the greatest weight is allocated to the cluster. Clusters have been identified by the network, but we know little about them. We will come back a little later to the topic of identifying clusters. For the output layer, the original SOMs used bidimensional grids. This was like an artifact of earlier research into identifying features of a two-dimensional array of pixel values in the composed images. The output layer can really have some structure — with three-dimensional neighborhoods described as a network of hexagons or anything laid out in some other way.

Chapter 8: Memory-Based Reasoning and Collaborative Data Filtering

You hear someone speaking and you automatically presume she's from Australia. Why? For what? Since her accent is reminiscent of other Australians you knew. Or try a new restaurant hoping to like it, because it was suggested by a friend with good taste. Both cases are examples of experiential decision making. When presented with new circumstances, human beings are motivated by memories of similar situations in the past. The data mining techniques presented in this chapter are based on that. Nearest neighboring strategies are based on the similitude principle. Memory-based reasoning (MBR) findings are based on previous equivalent situations — like agreeing that an Australian new friend is based on past examples of Australian accents. Collaborative filtering provides more detail, using not only the neighborhood similarities but also their preferences.

The suggestion to restaurant is an example of collective filtering. The principle of similitude is fundamental to all these strategies. What allows circumstances identical to a current scenario currently in the past? As well as locating the simi-lar records from the past, there is the challenge of integrating neighborhood knowledge. These are the two main principles for approaching closest neighbours.

This chapter starts with an introduction to MBR and how it works. Since distance and similarity measurements are important to nearest neighboring techniques, there is a section on distance metrics, including a discussion of distance value for data types, such as free text, that has no clear geometric understanding. MBR 's ideas are demonstrated by a case study demonstrating how MBR was used to apply keywords to news stories. The chapter then looks at collaborative filtering, which is a common approach for making recommendations, especially on the Internet. Collaborative filtering is often focused on nearest neighbours, but with a slight twist — instead of grouping restaurants or movies into neighbourhoods, it puts together the people who suggest them.

Memory Based Reasoning

The human capacity to reason from experience relies on the ability to recall suitable examples from the past. A doctor who detects illnesses, a claims investigator who flags false insurance claims and a mushroom hunter who finds Morels are all pursuing a similar procedure. Everyone first recognizes similar cases from experience and then applies to the problem at hand what their knowledge of those cases is. This is the basis of logic dependent upon experience. To locate preclassified records identical to a new record, a data-base of established records is scanned. These neighbors are used for evaluation and classification. MBR technologies cover many areas: identification of fraud. It is possible that new cases of fraud are identical to known cases.

For further research MBR will locate them and mark them. Predicting consumer response. The next clients likely to respond to an offer are possibly identical to previous clients who responded. MBR can quickly classify the clients who are likely to be next. Health counseling. For a particular patient the most appropriate treatment is undoubtedly the treatment that has resulted in the best results for similar patients. MBR is able to find the treatment which produces the best result. Responses are categorized. Free-text replies, for example on the U.S. Cen- sus for occupation and business, or grievances from cust omers, must be grouped into a set of fixed codes. The free-text MBR will process and allocate the codes. One of MBR 's strengths is its ability to use data "as is." Unlike other data mining methods, the format of the records doesn't matter. It is concerned only with the existence of two operations: a distance function capable of measuring a distance between any two records and a combination function capable of combining results from many neighbors in order to arrive at a response. For certain kinds of documents, these functions are readily established, including documents with complicated or unique data types such as geographic positions, photographs and free text, which are typically difficult to manage with other analytical techniques.

A case study later in the chapter demonstrates MBR's efficient application to news story classification — an example that takes advantage of the news story's full text to allocate topic codes. Another attribute of MBR is its adaptability. The simple introduction of new data into the historical database helps MBR to learn about new concepts and old meanings. MBR also provides good results without a long duration of preparation or the massaging of incoming data in the correct format.

Those benefits come at a price. MBR appears to be a resource abuser, since there must be a huge amount of historical data readily available to locate neighbours. Classifying new records which involve processing all historical records to find the most similar neighbors — a more time-consuming process than applying a neural network that has already been trained or a decision tree that has been developed. Finding good distance and combination functions is also a challenge, and also involves a bit of trial, error, and intuition.

Example: Use MBR in Tuxedo, New York to estimate rents

The purpose of this example is to demonstrate how MBR works by estimating the cost of renting an apartment in the target city by combining rent data in many similar cities — its nearest neighbours. MBR operates by first defining and then integrating the details from neighbours.

The first of those steps is shown in figure 8.1. The goal is to predict the city of Tuxedo in Orange County , New York by looking at its neighbours. Not its geographical neighbors along the rivers Hudson and Delaware, but its neighbors based on descriptive variables — in this case, popularity and average home value. The scatter plot shows cities in New York separated by those two variables. Figure 8.1 shows that Brooklyn and Queens are close neighbours, measured this way, and both are distant from Manhattan. While Manhattan is almost as populous as Brooklyn and Queens, its home prices alone place it in a band.

On the scatter plot shown in Figure 8.1 the first stage of MBR identifies the nearest neighbor. Then we find the next closest neighbor, and so on until the desired number is available. In this situation, the number of neighbors is two, and the closest ones turn out to be Shelter Island (which is actually an island) out by the tip of Long Island's North Fork, and North Salem, a town along the Connecticut border in North-ern Westchester. These towns sit at about the center of a population-sized list and close to the top of one sorted by home value. While they are several miles apart, Shelter Island and North Salem are very close to Tuxedo in these two dimensions. Upon position of the neighbors, the next step is to combine neighborhood information to infer something about the target. For this example, the intention is to estimate the cost of renting a Tuxedo house.

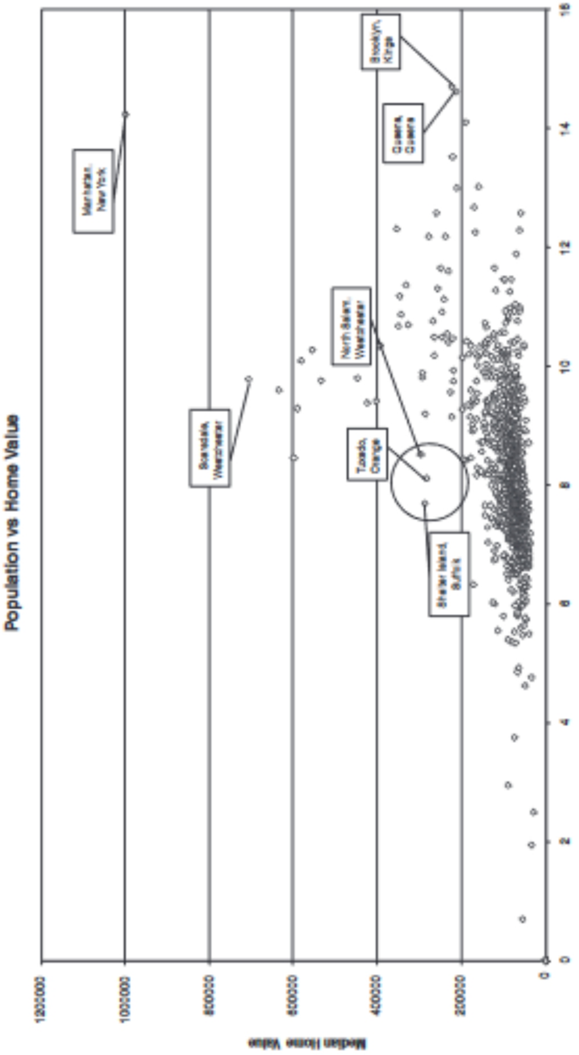


Figure 8.1 Based on 2000 census population and home value, the town of Tuxedo in Orange County has Shelter Island and North Salem as its two nearest neighbors.

There's more than one sensible way to combine neighboring results. The census includes two sources of information on rentals. Table 8.1 shows what the 2000 census says about rents in the two adjacent selected cities. In each of the many price ranges, there is a count of the number of households paying rent for each town as well as the median rent for each town. The challenge is to find out how to best use this data to classify neighborhood rents, and then how to incorporate neighborhood information to generate an estimate that similarly characterizes Tuxedo rents.

Table 8.1 The Neighbors

TOWN	POPULATION	MEDIAN RENT	RENT <$500 (%)	RENT $750 (%)	RENT $1500 (%)	RENT $1000 (%)	RENT >$1500 (%)	NO RENT (%)
Shelter Island	2228	$804	3.1	34.6	31.4	10.7	3.1	17
North Salem	5173	$1150	3	10.2	21.6	30.9	24.2	10.2

The nearest neighbors of Tuxedo, the towns of North Salem and Shelter Island, have very different rental distributions although the median rents are identical. A majority of homes in Shelter Island, 34.6 percent, rent the range from $500 to $750. The highest number of homes in the city of North Salem, 30.9 percent, leased from $1,000 to $1,500. In comparison, though only 3.1 per cent of Shelter Island homes pay more than $1,500, 24.2 per cent of North Salem homes do. On the other hand, at $804, Shelter Island's median rent is above the most common $750 ceiling, while North Salem's median rent of $1,150 is below the floor of the most common range for that area. If the average rent was available, it would also be a good candidate to describe the rents in the different cities.

One conceivable hybrid feature will be to compare the two neighbors' most regular rentals. Since there are only ranges available we use the mid-points. The midpoint of the most popular range, for Shelter Island, is $1,000. It's $1,250, for North Salem. Averaging the two leads to a $1,125 rent estimate at Tuxedo. Another mix feature will choose the midway point between the two midrange rents.

This second approach gives rent figures of $977 in Tuxedo. As it happens, there are a majority of rentals in Tuxedo within the range of $1,000 to $1,500 with the midpoint at $1,250. In Tuxedo the median rent is $907. Thus, median averaging marginally overestimates the median rent in Tuxedo and marginally underestimates the most common rent in Tuxedo by averaging the most common. It is impossible to tell which is better. The moral is that the obvious "right" combination feature doesn't always exist.

The Challenges of MBR

The training set consisted of all New York cities in the simple example provided, each represented by a handful of numerical fields such as population, median home value, and median rent. The distance was calculated by putting on a scatter plot with axes scaled to reasonable ranges and arbitrarily limiting the number of neighbors to two. The mixing feature was a simple average. These all seem fair choices. Using MBR usually requires many choices:

1. Select a suitable set of Training records

1. Choosing the most appropriate means of describing the training records.

1. Selecting the distance function, combination function, and several neighbors, Let us look at each of these in turn.

Choosing a Balanced Set of Records

The Training Collection is a compilation of records from history. It needs to have good population coverage to make it useful for statistical purposes to the nearest neighbors of a new database. For all values, a random sample cannot have adequate coverage. Certain categories are more frequent than others, and the random sample is dominated by more frequent categories. For example, fraudulent transactions are much rarer than non-fraudulent transactions; the cardiac disease is much more common than liver cancer, computer industry news reports more common than plastics, and so on. If necessary, the training set should include approximately equal numbers of records representing the various categories to maintain consistency.

Representing Training Data

MBR's success in making predictions depends on how you view the training set. The scatter plot method shown in Figure 8.2 works for two or three variables and a limited number of documents, but it is not well-scale. The easiest way to locate nearest neighbors involves determining the distance from the unknown case to each record in the training set and selecting the smallest distance training records. If the number of records increases, the time it takes to locate the neighbors for a new record is increasingly growing. This applies especially if the records are stored in a relational database. In this case, the question looks like:

```
 SELECT distance(),rec.category
FROM historical records rec
ORDER BY 1 ASCENDING;
```

The notation distance() stands in for whatever distance function appears to be. In this situation, all historical documents need to be filtered to get the handfuls needed for neighbors closest to them. This includes a full-table scan plus a kind of — quite a few costly operations. Eliminating the sort can be achieved by walking through the table and holding another table nearest to it, adding and removing records as necessary. Unfortunately, without the use of a procedural language, this method is not readily expressible in SQL.

Relational database efficiency is reasonably good nowadays. The challenge with MBR scoring data is that it is important to compare each case being scored to each case in the database. It doesn't take long to score a single new record, even though there are millions of historical records. However, will setting lots of new records have bad results. Another way of making MBR more successful is by reducing the number of records in the training collection.

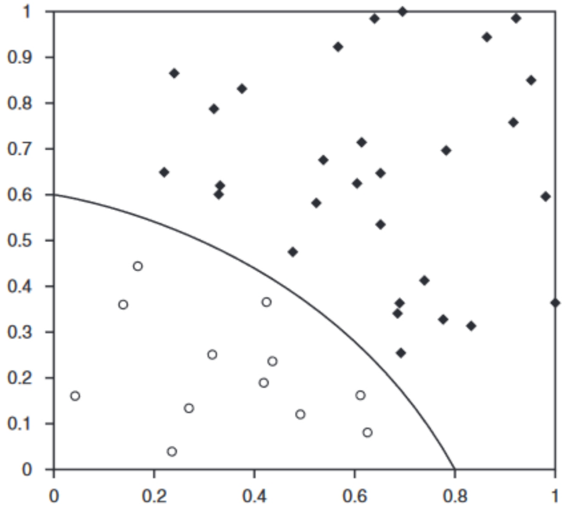


Figure 8.2 Perhaps the cleanest training set for MBR is one that divides neatly into two disjoint sets.

Figure 8.2 displays categorical data with a scatter plot. This graph has a fixed boundary between the two regions. The diamonds are the points above the line, and the ones below the line are all circles. While there are forty points in that graph, most of the points are redundant. That is, they are not really required for the purposes of classification.

Figure 8.3 shows that to achieve the same results, only eight points are needed in it. Given the size of the training set has such an important impact on MBR performance being able to reduce the size is a major boost in performance. How do you find the reduced collection of records? The most realistic method is to search for clusters that contain records from different categories. The cluster centers can then function as a reduced group. This works well because the various categories are very different.

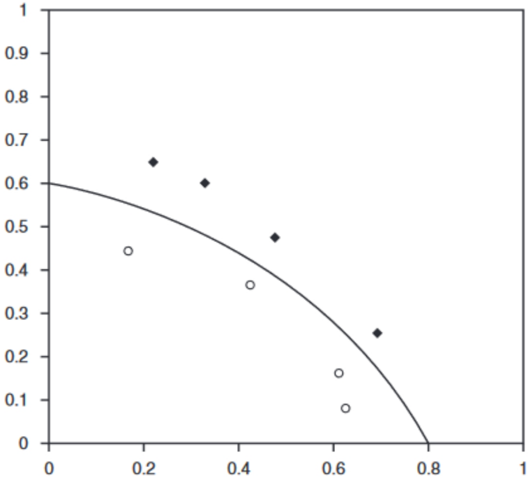


Figure 8.3 This smaller set of points returns the same results as in Figure 8.2 using MBR.

However, where there is a certain overlap, and the categories are not so well-defined, the use of clusters to reduce the size of the training set can trigger poor results for MBR. One field of recent research has been seeking an ideal collection of "service records." When such an ideal collection can be found, the historical records can often be reduced to the level they fit within a spreadsheet, making applying MBR to new records on less efficient machines very successful.

Determination of Distance Functions, Combination Functions, and Neighbor Numbers

The main ingredients in using MBR are distance function, combination function, and a number of neighbors. Depending on these criticalities, the same collection of historical records will prove to be very helpful or not at all for predictive purposes. Luckily, simple distance functions and combination functions work generally very well. Let's look at a small case study before addressing those issues in depth.

Distance Measuring

Say your travels will take you to a small town and you want to know what the weather is like. If you have a newspaper reporting weather forecasts for major cities, usually what you'd do is find the forecast for cities near the small town. You could be looking at the nearest city and just taking its weather, or making some sort of combination of forecasts for, say, the three nearest cities. This is an example of using MBR to find forecast weather. The distance feature to use is the geographic distance between the two places. Online sites that include a weather forecast for any zip code given by a user are likely to do something similar.

What is a distance Function?

Distance is a way similarity is calculated by the MBR. The distance measure from point A to point B, denoted by d(A, B), for any true distance metric has four main properties:

1. **Identified well:** The distance between two points is always set and is a real non-negative integer, $d(A,B) \geq 0$.

2. **Identity:** The distance to itself from one point is always zero, so $d(A, A) = 0$

3. **Commutativity:** There is no difference in direction, so the distance from A to B is the same as the distance from B to A: $d(A, B) = d(B, A)$. This house, for example, precludes single-way roads.

4. **The Triangle Inequality:** On the way from A to B, visiting an intermediate point C never shortens the gap, so ALB $d(A,B) \geq d(A,C) + d(C,B)$.

The points are, for MBR, really records in a database. This formal description of distance is the basis for calculating similarity, but when some of these restrictions are relaxed a bit, MBR still works pretty well. The distance function in the case study of the classification of news stories, for example, was not commutative; that is, the distance from a news story A to another B was not always the same as the distance measure from B to A. But the calculation of similarity was still useful for classification purposes.

What makes MBR useful with those properties? The fact that distance is well-defined means that every record has a neighbor in the database somewhere — and MBR requires neighbors to function. The identity property makes distance adhere to the intuitive notion that the original record itself is the most comparable record to a given record. Commutativity and the Triangle Inequality render local, and well-behaved neighbors close by. The inclusion of a new record into the database would not put an existing record forward.

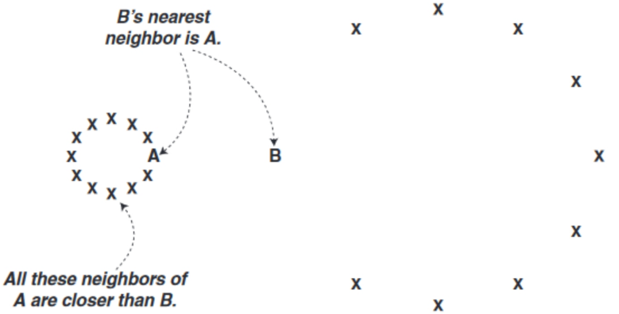

B's nearest neighbor is A.
A
B
All these neighbors of A are closer than B.

Figure 8.5 B's nearest neighbor is A, but A has many neighbors closer than B.

The similarity is a matter reserved for just two albums at a time. Although the distance measure used to locate nearest neighbors is well-behaved, some unusual properties may be present in the set of nearest neighbors. For example, the closest neighbor to a record B may be A, but A may have several closer neighbors than B, as shown in Figure 8.5. This condition does not raise MBR issues.

Building a Distance Function

One Field at a Time Distance as a geometric term is simple to understand, but how can distance be specified for records consisting of several different fields of different types? The response is one area at a time. Consider some details of the samples, as shown in Table 8.7.

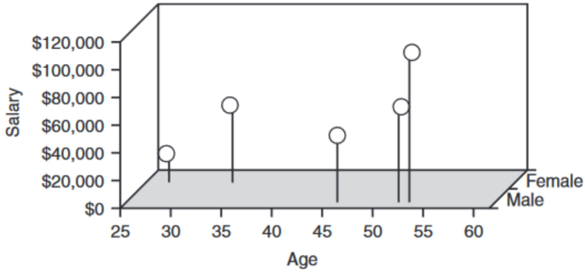


Figure 8.6 This scatter plot shows the five records from Table 8.7 in three dimensions—age, salary, and gender—and suggests that standard distance is a good metric for nearest neighbors.

Figure 8.6 illustrates a three-dimensional Scatter Graph. The records are a little confusing, with two quantitative and one categorical category. This example demonstrates how to define field distance functions for each field and then combine them into a single record distance function, which gives two records a distance.

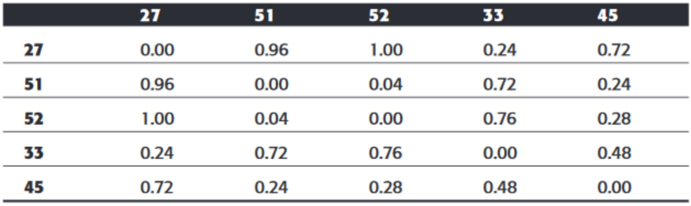
Table 8.7 Five Customers in a Marketing Database

RECNUM	GENDER	AGE	SALARY
1	female	27	$ 19,000
2	male	51	$ 64,000
3	male	52	$105,000
4	female	33	$ 55,000
5	male	45	$ 45,000

For numeric fields, the four most common distance functions are:

- Absolute value difference :|A–B|
- Difference square: $(A - B)^2$
- Normalized absolute value: |A–B|/(max difference)
- Absolute value of difference (standardized values): |(A – mean)/(standard deviation) – (B – mean)/(standard deviation)| which is equivalent to |(A – B)/(standard deviation)|.

Table 8.8 Distance Matrix Based on Ages of Customers



	27	51	52	33	45
27	0.00	0.96	1.00	0.24	0.72
51	0.96	0.00	0.04	0.72	0.24
52	1.00	0.04	0.00	0.76	0.28
33	0.24	0.72	0.76	0.00	0.48
45	0.72	0.24	0.28	0.48	0.00

The benefit of the absolute normalized value is that it's always between 0 and 1. Since ages in this example are much smaller than incomes, the normalized absolute value is a good choice for both of them — so no area can dominate the record distance function (difference of standardized values is a good choice as well). For ages, the matrix of distance appears as Table 8.8.

Memory-based reasoning is a popular technique in data mining that can be used to solve a wide range of data mining issues involving classification or estimation. Unlike other data mining techniques which use a pre-classified data training set to construct a model and then discard the training set, for MBR, the training set is essentially the model. Perhaps the most important step in MBR is choosing the right set of instructions. The training set needs to contain all possible classifications in a reasonable number of examples. This may mean enriching it by adding a large number of uncommon classification instances to create a balanced train-set of approximately the same amount of instances across all categories.

A training set that only contains instances of poor customers will assume that all customers are poor. In general, the scale of the training set should include examples from at least thousands, if not hundreds of thousands or millions. MBR is an approach k-nearest to neighbors. This includes a distance function to decide which neighbors are close. Many methods are used to determine the difference between the two documents. A crucial step in using MBR is the careful selection of an acceptable distance function.

The chapter presented an approach to generating an overall distance function by constructing and normalizing a distance function for each field. The normalized field distances can then be combined or summed up in a Euclidean fashion to create a distance to Manhattan. By using the Euclidean method, a large difference is necessary for anyone area to cause two records to be considered far apart. The Manhattan approach is more forgiving — a major difference in one field can be compensated more easily by smaller values on other fields.

A validation collection can be used to choose the best distance function for a given set of models by applying all candidates to see which results better. The right choice of neighbors often depends on changing the distance function to favor some fields over others. Incorporating weights into the distance feature makes this easy to achieve. The next question is on how many neighbors to choose from. Once again, using the validation set to invest different numbers of neighbors will help to decide the optimal number. There is no proper neighboring number. The number is dependent on the data distribution and is strongly dependent on the problem being solved. Using weights inversely proportional to distance, the simple combination method, weighted voting, does a good job for categorical results. The equivalent method is a weighted average for calculating statistical values. Recommendations are one strong method for memory-based reasoning. Collaborative filtering is a method for making suggestions that work by putting together people with similar tastes using a distance feature that can compare two lists of user ratings. Recommendations for a new person are determined using a weighted average of his / her closest neighbors' scores.

Chapter 9: Association Rules and Market Basket Analysis

Starting with the concept of the shopping cart in Figure 9.1 filled with almost every products purchased by someone on a quick trip to the supermarket, to convey the fundamental ideas of market basket analysis. This basket contains a product range — orange juice, bananas, soft drink, window cleaner, and detergent. One basket tells us what one customer had bought at one time. A complete list of all customer purchases provides much more information; it describes the most important part of a retail business — what merchandise customers buy, and when. At different times each customer buys a different set of products in different quantities. Analysis of market baskets uses information about what customers buy to provide insight into who they are and why they make purchases. Analysis of consumer baskets offers insight into the goods by telling us which items appear to be bought together and which are most appropriate for promotion.

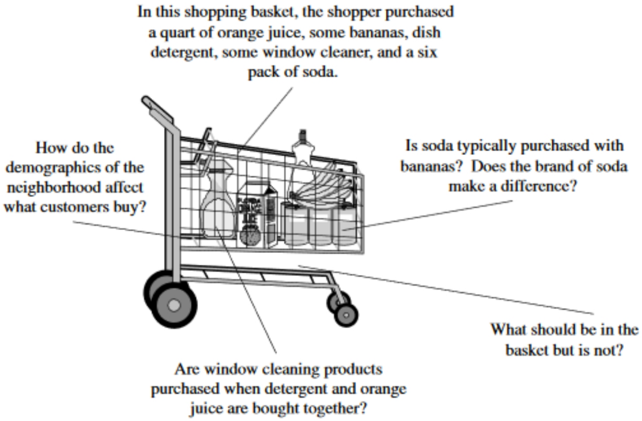

In this shopping basket, the shopper purchased a quart of orange juice, some bananas, dish detergent, some window cleaner, and a six pack of soda.
How do the demographics of the neighborhood affect what customers buy?
Is soda typically purchased with bananas? Does the brand of soda make a difference?
What should be in the basket but is not?
Are window cleaning products purchased when detergent and orange juice are bought together?

Figure 9.1 Market basket analysis helps you understand customers as well as items that are purchased together.

This information can be actioned: it can suggest new layouts of the store; it can determine that which products to put on special; it can indicate when to issue coupons, etc. It becomes even more valuable when this data can be linked to individual customers through a loyalty card or website registration. The most closely related data mining technique to market basket analysis is automatic association rule generation. The rules of the association represent patterns in the data without a given target. As such, they are an undirected example of data mining. Whether the patterns make sense was left to the rules of the Association of Human Interpretation originally derived from point-of-sale data describing what products are purchased together. Although its roots lie in analyzing point-of-sale transactions, association rules can be applied outside the retail sector to find relationships among other types of "baskets."

Items purchased on a credit card, such as rental cars and hotel rooms, give insight into the further next product that customers are likely to buy.

Optional services can purchased by telecommunications customers (call waiting, call forwarding, DSL, speed call, etc.) help to determine how these services can be bundled together to maximize revenue.

Retail customer banking services (money market accounts, CDs, investment services, car loans, etc.) identify customers who are likely to want other services.

Odd combinations of insurance claims may be a sign of fraud and may lead to further investigation.

The medical history of patients may provide indications of probable complications based on certain treatment combinations. Even the rules of association do not live up to standards. For example, in our experience, they aren't a good choice to build cross-selling models in industries like retail banking, because the rules end up describing previous marketing promotions.

In retail banking as well, usually, customers start with a checking account and then a savings account. Differentiation between products doesn't occur until more products are available to consumers.

This chapter discusses both the risks and association rules usages. The chapter start or begin with an overview of the market basket analysis, including more basic market basket data analyzes, which do not require association rules. Then it falls into laws of association, describing how they are formed. The chapter then continues with ways of extending association rules to include other aspects of the analysis of the market basket.

9.1 Defining An Analysis of Market Baskets

Market basket analysis doesn’t refer to a single technique; it refers to a set of business issues related to the understanding of transaction data from the point of sale. Association rules are the most common technique, and much of this chapter delves into that subject. This section talks about market basket data before talking about the association rules.

Three levels of basket data for the Market Basket Data Market are transaction data describing three fundamentally different entities:

- **Customers**
- **Orders (also known as transactions or baskets)**
- **Items**

The data structure for market basket data in a relational database often looks similar to Figure 9.2. This data structure is composed of four major entities.

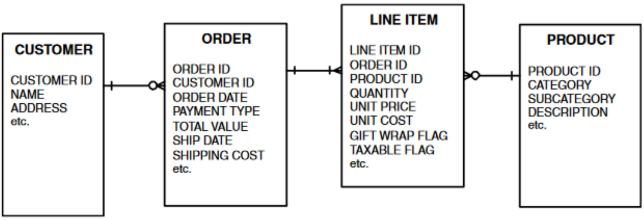

CUSTOMER
CUSTOMER ID
NAME
ADDRESS
etc.
ORDER
ORDER ID
CUSTOMER ID
ORDER DATE
PAYMENT TYPE
TOTAL VALUE
SHIP DATE
SHIPPING COST
etc.
LINE ITEM
LINE ITEM ID
ORDER ID
PRODUCT ID
QUANTITY
UNIT PRICE
UNIT COST
GIFT WRAP FLAG
TAXABLE FLAG
etc.
PRODUCT
PRODUCT ID
CATEGORY
SUBCATEGORY
DESCRIPTION
etc.

Figure 9.2 A data model for transaction-level market basket data typically has three tables, one for the customer, one for the order, and one for the order line.

The order is the basic data structure for data from market baskets. An order represents a single event of a client's purchase. This could correspond to a customer ordering multiple products on a website or a customer pursuing a food basket or a customer purchasing multiple items from a catalog. This includes the total purchase amount, total amount, additional shipping charges, type of payment, and any other transaction data relevant. A unique identifier is occasionally given to the transaction. Sometimes you have to cobble the unique identifier from other data together.

In one example, we needed to combine four fields to get a purechase identifier in a store — the time mark when the customer paid, chain identification, store identification, and lane identification. Individual items in the order are separately shown as line items. These data include the price paid for the item, the number of items, whether tax should be charged, and possibly the cost (which can be used to calculate the margin). Typically the item table also has a link to a product reference table providing more descriptive information about each product.

This descriptive information should include the hierarchy of the product and other information, which may prove valuable for analysis. The customer table is an optional table and should be available if, for example, a customer can be identified on a website requiring registration or when the customer is using an affinity card during the transaction.

While the customer table may have interesting fields, the most powerful element is the ID itself, because it can link transactions over time together. For example, tracking customers over time allows us to determine which grocery shoppers "bake from scratch"—something of keen interest to flour makers as well as prepackaged cake mixes. Such customers could be identified by the frequency of their purchases of baking powder, flour, and similar ingredients, the proportion of such purchases to the total customer expenditure, and the lack of interest in prepackaged mixes and ready-to-eat desserts. Such ingredients can, of course, be purchased at different times and in different quantities, making it necessary to tie multiple transactions over time together. All three basket data levels for the market are important. There are some basic measures, for example, to underpin orders:

What is the average number of customer orders?

- What is the average number or amount of products per order?
- An average number of items per order?
- What is the proportion or balance of customers who have ever bought the product?
- What is the average quantity of orders per customer included in the item for a given product?
- What is the average quantity purchased in an order when the product is purchased for a given product? These indicators provide broad insight into the company.

There are few repeat customers in some cases, so the percentage of orders per customer is close to 1; this suggests a business opportunity to increase the number of customer sales. Or the number of products per order could be close to 1, suggesting an opportunity to cross-sell during the order process. Comparing those measures with each other can be useful. We found that the number of orders is often a useful way to differentiate between customers; good customers clearly order more frequently than not-so-good clients.

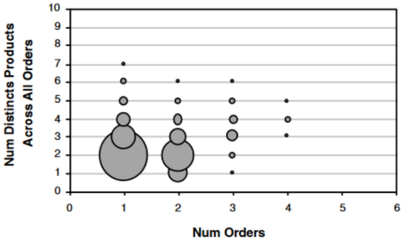


Figure 9.3 This bubble plot shows the breadth of customer relationships by the depth of the relationship.

Figure 9.3 tries to look at the breadth of the customer relationship (the number of unique items ever bought) by the depth of the relationship (the number of orders) for customers who bought more than one item. This data comes from a tiny specialty retailer. The biggest bubble shows that many customers who buy two products do so simultaneously. A surprisingly large bubble also shows that a considerable number of customers buy the same product in two orders. Better customers, those who have returned multiple times-tend to buy more diverse goods. But some of them return and buy the same thing they first bought. How can the retailer encourage customers to come back and purchase more and various products? Analysis of market baskets can't answer the question, but it can at least motivate you to ask it, and maybe give hints that might help.

Characteristics of Order

Additional notable characteristics include consumer transactions. The average order size, for example, varies by time and region — and it's important to keep track of these to understand shifts in the market climate. In reporting systems, such information is also accessible because it is conveniently summarized.

However, certain information may need to be gleaned from the data at the transaction level. Figure 9.4 breaks down purchases with another store by the order size and the credit card used to pay –

Visa, MasterCard, or American Express. The first thing that need to remember is that whichever credit card is used, the larger the order, the greater the average purchase price. That would be helpful. Furthermore, the use of one form of credit card, American Express, is frequently correlated with larger orders — an interesting finding of those clients.

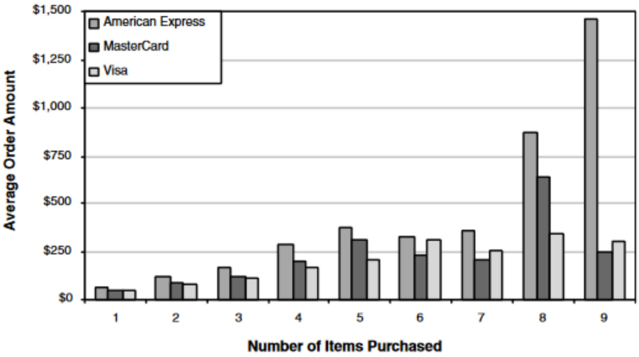


Figure 9.4 This chart shows the average amount spent by credit card type based on the number of items in the order for one particular retailer.

Additional information can also be obtained at the point of sale for Online sales and mail-order transactions:

- Has the order been using a gift wrap?
- Is the order ships to the same address as the billing address?
- Did the buyer approve or reject a specific bid for cross-selling?

Of course, collecting information and making it available for review at the point of sale are two separate items. Gift giving and openness to cross-selling deals, however, are two very useful things about clients to remember. Finding trends with this information involves first collecting the information (at the call center, or through the online interface) and then transferring it to a data-mining environment.

Popularity of Item

Which articles are the most popular? Typically, this is a question that can be addressed by looking at inventory curves that can be generated without having to deal with data at the transaction level. Knowing the sales of a single item is just the beginning, however. Linked questions are:

- What is the most common item in a single-item order?
- Which object is the most popular in multi-item commands?
- What is the most popular thing among repeat buyers customers?
- Over time, how has the popularity of unique products changed?
- How does an item's popularity vary from region to region?

The first three questions are especially important as they may provide suggestions for increasing customer relationships. These questions can be answered by association rules, particularly when used with virtual products to reflect the size of the order or the number of orders that a customer has made. The last two questions discuss the time and geography dimensions that are very relevant for business basket research applications. In different regions, different items have different affinities — something that retailers are very familiar with. It is also possible to use association rules, by adding virtual items for region and seasonality, to begin to understand these areas.

Tracking Marketing Responses or Interventions

Looking overtime at the individual items will provide a clear picture of what happens to the product. The impact of the interventions can be shown by using marketing strategies along with product sales over time, as seen in Figure 9.5. The diagram shows a sales curve for a given commodity. Sales are running at 50 units a week prior to the intervention. They peak at about seven or eight times that amount after the intervention, before slipping slowly down over the six or seven weeks. The reaction of the marketing campaign can be calculated using such charts.

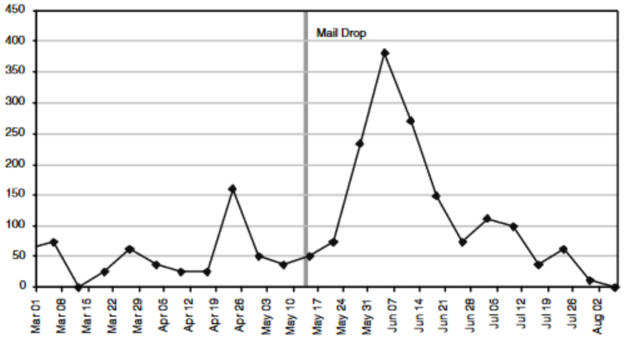


Figure 9.5 Showing marketing interventions and product sales on the same chart makes it possible to see effects of marketing efforts.

Such research does not need to look at individual consumer baskets-regular, or weekly product sales summaries that are appropriate. However, it needs awareness when marketing campaigns take place — and the biggest obstacle often is having such a calendar. One of the essential questions which can be answered by such a chart is the intervention effect. The difficulty in responding to this question is to decide whether additional purchases are incremental or are made at some later time by consumers who would buy the product anyway.

Data from the market basket will begin to address that question. Besides looking at the sales volume after an intervention, we can also look at the number of baskets containing the object. If there is no rise in the number of customers, there is proof that current customers simply store at a lower cost on the item. A similar concern is whether discounting leads to further sales of other goods. Rules of association can help to answer this question by finding combinations of products that include those marketed during the promotion era. Likewise, we would want to know whether the average order size increases or decreases following an operation. These are examples of questions where more precise data on the transaction level is needed.

Clustering of products by usage

Perhaps one of the fascinating questions is what product types always turn up together. Such product groups are very useful in making recommendations to customers — customers who have bought some of the products may be interested in the rest of them. In this region, the rules of association provide some answers at the individual product level. In particular, this data mining technique decides which product or products at a purchase simultaneously indicate the purchase of other unique items. Often it is desirable to have larger clusters than those given by rules on the association, which contain just a few things in each rule.

Standard clustering techniques can also be used on market basket data. In this example, as shown in Fig 9.6, the data must be rotated so that each row represents one order or consumer, and for each product purchased, there is a flag or counter. Unfortunately, thousands of different items also come in. Such a transition should take place at the category level, rather than at the individual product level, to minimize the number of columns. Usually, there is a lot of product details available. Besides the product hierarchy, other details include clothing color, whether food is low calorie, whether a poster contains a frame, etc. Such definitions provide a wealth of knowledge and can lead to useful ad hoc questions:

- Tendentially selling food items together?
- Do consumers purchase similar clothing colors at the same time?
- Do consumers who purchase framed posters buy additional items as well?

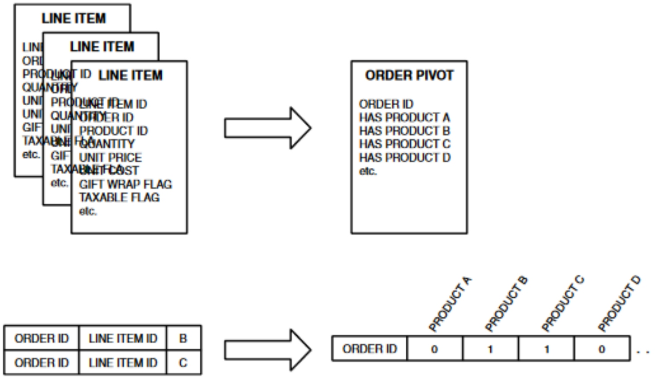

LINE ITEM
LINE ITEM
LINE ITEM
LINE ITEM ID
ORDER ID
PRODUCT ID
QUANTITY
UNIT PRICE
UNIT COST
GIFT WRAP FLAG
TAXABLE FLAG
etc.
ORDER PIVOT
ORDER ID
HAS PRODUCT A
HAS PRODUCT B
HAS PRODUCT C
HAS PRODUCT D
etc.
ORDER ID
LINE ITEM ID
B
ORDER ID
LINE ITEM ID
C
PRODUCT A
PRODUCT B
PRODUCT C
PRODUCT D
ORDER ID
0
1
1
0

Figure 9.6 Pivoting market basket data makes it possible to run clustering algorithms to find interesting groups of products.

It is also more useful to be able to answer such questions than to pursue cluster goods, as such guided questions also lead directly to marketing behavior.

Rules of Association

One advantage of association rules is the consistency and usefulness of the outcomes, which are in the form of product group laws. An association rule is appealed intuitively since it explains how concrete goods and services are grouped together. A rule like, "if a customer buys three-way calling, then that customer will buy call waiting as well," is simple. Perhaps better, a particular plan of action might be suggested, such as bundling three-way calling with call waiting into a single service bundle. While the rules of the association are easy to understand, they aren't always helpful. Examples of true rules created from real data are the following three laws:

Customers who buy Barbie dolls from Wal-Mart also have a 60% chance of buying one of three styles of candy bars.

It is very likely that consumers who sign maintenance agreements will buy large appliances.

When a new hardware store opens, toilet bowl cleaners are one of the most widely sold products. The last two examples are examples we actually saw in the results. The first is an example that was quoted on 8 September 1997 in Forbes. These three analyses demonstrate the three common types of rules created by the rules of association: the actionable, the trivial, and the incomprehensible. Apart from these types of rules, the "Famous Rules" sidebar also talks about another group.

Guided and Actionable Rules

The useful rule contains information of high quality that can be used. It is also not hard to explain once the pattern is identified, and telling a story can lead to observations and action. Barbie dolls, who prefer chocolate bars to other kinds of food, is not a probable reality. Imagine a family going out instead. The aim: to find a gift for Emily, a friend of little Susie, because her birthday is coming up. The best present is a Barbie doll. Little Jacob starts crying at check-out. He also wants something — a candy bar that fits the bill. Or maybe Emily has a brother; the gift-giving festivities cannot leave him out. Perhaps the candy bar is for Mom, as buying Barbie dolls is a tiresome task, and Mom needs some energy.

These examples all say the candy bar is an impulse buy added to the Barbie dolls. It is not clear if Wal-Mart will make use of that knowledge. This rule can recommend more prominent product placement on their way back from Barbie-land, such as ensuring customers have to walk through candy aisles. It can recommend product tie-ins and promotions that combine to sell candy bars and dolls. This may recommend new ways of advertising goods. It suggests probable causes and potential solutions since the rule is easily understood.

Trivial Regulations

Anyone at all acquainted with the company already knows trivial outcomes. The second example ("Customers who buy maintenance contracts are highly likely to buy large appliances") is an example of a trivial law. In reality, at the same time, customers usually buy maintenance agreements and large appliances. Why else will they snap up maintenance contracts? The two are marketed together and seldom sold separately (though it is the big appliance that is sold without the agreement, rather than the offer sold without the appliance when sold separately).

However, this law was noticed after hundreds of thousands of Sears point-of-sale transactions were analyzed. Despite being legitimate and well supported in the evidence, it is all in vain. Similar findings abound: people buying 2-by-4s also buy nails; consumers buying paint buy paintbrushes; oil and oil filters are bought together, as are hamburgers and hamburger buns, and charcoal, and lighter fluid. The same category falls under a subtler concern. A seemingly surprising finding — such as the fact that people on their local telephone service who purchase the three-way calling option almost always purchase call waiting — may be the outcome of previous marketing campaigns and product packages. In case of telephone service options, three-way calling is usually combined with call waiting, so ordering it separately is difficult.

The study does not yield actionable findings in this case; it produces results that have already been acted upon. While it poses a risk to any data mining technique, market basket analysis is especially susceptible to replicate the performance of previous marketing campaigns due to its reliance on unresumed point-of-sale data — the very same data that determine the campaign's success. Business basket research findings will also assess the effectiveness of previous marketing campaigns. There is one use of trivial rules, but it is not explicitly used in data mining. When a rule can appear 100 percent of the time, the few cases it does not provide a great deal of data quality knowledge. In other words, the exceptions to trivial rules point to areas where business processes, data collection, and analysis can need to be refined further.

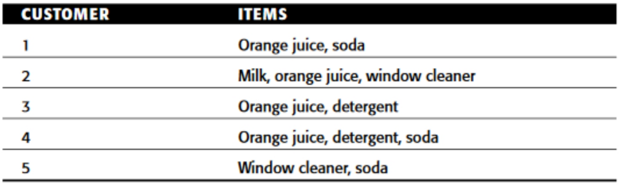
Inexplicable Rules

Inexplainable findings do not seem to have any cause and do not indicate any course of action. The third trend ("When a new hardware store opens, one of the most com-monthly sold products is toilet bowl cleaner") is interesting, enticing us with a new reality while offering details that do not offer insight into customer behavior or product or imply additional actions. In this situation, the trend for new store openings was discovered by a major hardware corporation, but they could not find out how to benefit from it. During the store openings, several things are on sale, but the toilet bowl cleaners stood out. Any clarification could be provided by further research: Is the discount on toilet bowl cleaners much greater than for other products? Are they regularly put for store openings in a high-traffic area, but other times have hidden? Is the outcome an anomaly in a handful of stores? Other times, are they difficult to find? Whatever the cause, it is unlikely that a plausible explanation can be provided by further analysis of only market basket data.

How Strong Is Rule of Association?

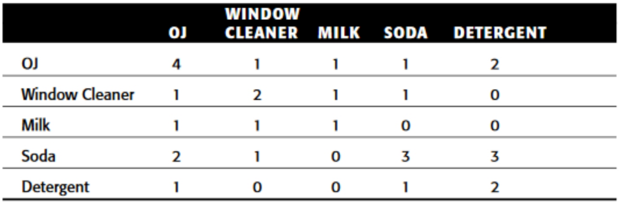
The rules of association begin with transactions containing one or more goods or service offers and some basic transaction details. The goods and services offered are called objects for research purposes. Table 9.1 describes five purchases at a shop holding five items. These transactions were streamlined to include only purchased products in this chapter, how to use details such as date and time, and whether the customer who paid with cash or a credit card is discussed later.

Table 9.1 Grocery Point-of-Sale Transactions



CUSTOMER	ITEMS
1	Orange juice, soda
2	Milk, orange juice, window cleaner
3	Orange juice, detergent
4	Orange juice, detergent, soda
5	Window cleaner, soda

Table 9.2 Co-Occurrence of Products



	OJ	WINDOW CLEANER	MILK	SODA	DETERGENT
OJ	4	1	1	1	2
Window Cleaner	1	2	1	1	0
Milk	1	1	1	0	0
Soda	2	1	0	3	3
Detergent	1	0	0	1	2

And of these purchases provides us with details on which goods to buy with various other goods. This is shown in a table of co-occurrence, which tells how many times each pair of products that was purchased together (see Table 9.2). The box where the row of "Soda" intersects the "OJ" column, for example, has a value of "2," indicating two transactions involve both soda and orange juice. This is easily checked against the original transaction records, where both products were purchased by Customers 1 and 4. The values along the diagonal (for example, the value in the column "OJ" and the "OJ" row) reflect the number of transactions that contain that object.

This basic table of coincidences already shows some common patterns:

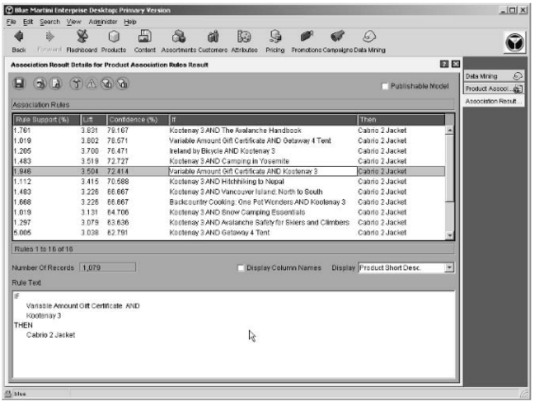
It is more likely to buy orange juice and soda together than any other two products.

- With window cleaner or milk, Detergent is never purchased.
- Milk is never bought with soda or detergent.

These statements are examples of correlations and might imply a specific rule such as: "If a customer buys soda, then the customer also buys orange juice." For now, let's postpone the discussion about how to automatically locate the law, and instead ask another query. What is this law useful for? Two of the five transactions in the data contain both soda and orange juice. These two transactions back the law. The rule is followed by two out of five or 40 percent. Because both the soda-containing transactions also contain orange juice, the rule also has a high degree of trust. In fact, two of the three soda-containing transactions also contain orange juice, so the "if soda, then orange juice" rule has 67 percent confidence.

The inverse law, "if the juice is green, then soda," has lower confidence. Of the four orange juice transactions, only two have a soda as well. His trust is just 50 percent, then. More formally, trust is the ratio of the number of transactions that support the rule to the number of transactions in which the conditional part of the rule holds. Another way to mean this is that trust is the ratio of number of transactions with all the items or details to the number of transactions with only the items "if." Another question is how much better the rule is than chance.

One way to answer this is to measure the lift (also called improvement), which tells us how much better a rule is to predict the outcome than just guess the outcome first. Lift is the ratio of target density after the left-hand side is added to the target density within the population. Another way to say this is that lift is the ratio of records supporting the whole rule to the number that would be predicted, provided there is almost no relationship between the products. A related calculation, the surplus, is the difference between the number of records that the whole law supports, minus the estimated value. Since the surplus is measured in the same units as the initial profits, dealing with this is often simpler. Figure 9.7 offers an example of the boost, trust, and help offered by Blue Martini, a company specializing in retail tooling. Their software framework contains a suite of analytical methods that incorporates rules of the association.


Blue Martini Enterprise Desktop: Primary Version
File
Edit
Search
View
Administer
Help
Back
Forward
Flashboard
Products
Content
Assortments
Customers
Attributes
Pricing
Promotions
Campaigns
Data Mining
Association Result Details for Product Association Rules Result
☐ Publishable Model
Association Rules
Data Mining
Product Assoc...
Association Result...
Rule Text
blue

Rule Support (%)	Lift	Confidence (%)	If	Then
1.761	3.831	79.167	Kootenay 3 AND The Avalanche Handbook	Cabrio 2 Jacket
1.019	3.802	78.571	Variable Amount Gift Certificate AND Getaway 4 Tent	Cabrio 2 Jacket
1.205	3.700	76.471	Ireland by Bicycle AND Kootenay 3	Cabrio 2 Jacket
1.483	3.519	72.727	Kootenay 3 AND Camping in Yosemite	Cabrio 2 Jacket
1.946	3.504	72.414	Variable Amount Gift Certificate AND Kootenay 3	Cabrio 2 Jacket
1.112	3.415	70.588	Kootenay 3 AND Hitchhiking to Nepal	Cabrio 2 Jacket
1.483	3.226	66.667	Kootenay 3 AND Vancouver Island: North to South	Cabrio 2 Jacket
1.668	3.226	66.667	Backcountry Cooking: One Pot Wonders AND Kootenay 3	Cabrio 2 Jacket
1.019	3.131	64.706	Kootenay 3 AND Snow Camping Essentials	Cabrio 2 Jacket
1.297	3.079	63.636	Kootenay 3 AND Avalanche Safety for Skiers and Climbers	Cabrio 2 Jacket
5.006	3.038	62.791	Kootenay 3 AND Getaway 4 Tent	Cabrio 2 Jacket

Rules 1 to 16 of 16

Number Of Records: 1,079

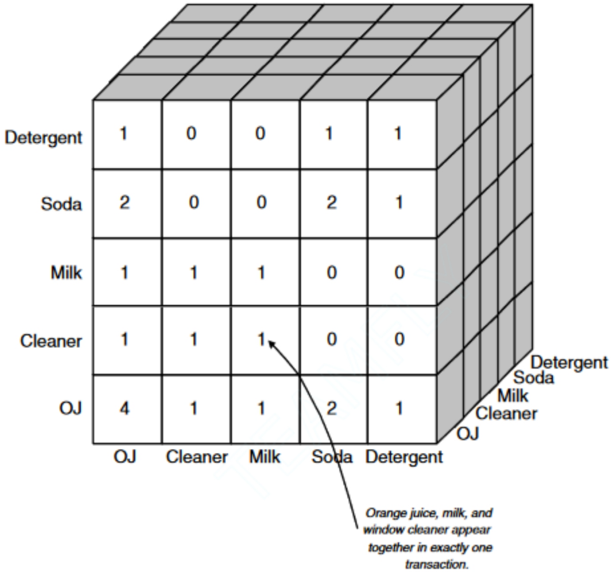
☐ Display Column Names

Display: Product Short Desc.

```
IF
    Variable Amount Gift Certificate AND
    Kootenay 3
THEN
    Cabrio 2 Jacket
```

Figure 9.7 Blue Martini provides an interface that shows the support, confidence, and lift of an association rule.

This specific example shows that a specific jacket with a gift certificate is much more likely to be purchased, a knowledge that can be used for enhancing communications to sell both gift certificates and jackets. The concepts behind the co-occurrence table apply to combinations, not just pairs of items, of any number of objects. Imagine a small cube with each side split into five different sections for combinations of three products, as shown in Figure 9.8. Although the data contains only five objects, there are already 125 separate subcubes to fill in.



	OJ	Cleaner	Milk	Soda	Detergent
Detergent	1	0	0	1	1
Soda	2	0	0	2	1
Milk	1	1	1	0	0
Cleaner	1	1	1	0	0
OJ	4	1	1	2	1

Orange juice, milk, and window cleaner appear together in exactly one transaction.

Figure 9.8 A co-occurrence table in three dimensions can be visualized as a cube.

This can be further reduced a bit (by a factor of six) by playing with symmetries in the cube, but the number of subcubes for groups of three items is proportional to the third power of the number of different items. The number of combinations of n items is generally proportional to the number of items raised to the nth power — a number that becomes very high, very quickly. And for each of these combinations, creating the co-occurrence table involves doing some work.

Creating Rules of Association

Figure 9.9 shows this basic method for discovering association rules. There are three critical issues when designing association rules:

Select the right set of objects.

Fixing rules by deciphering the numbers in the matrix of co-occurrence.

Overcoming the practical constraints imposed by thousands or tens of thousands of products. The next three parts look in more depth at these issues.

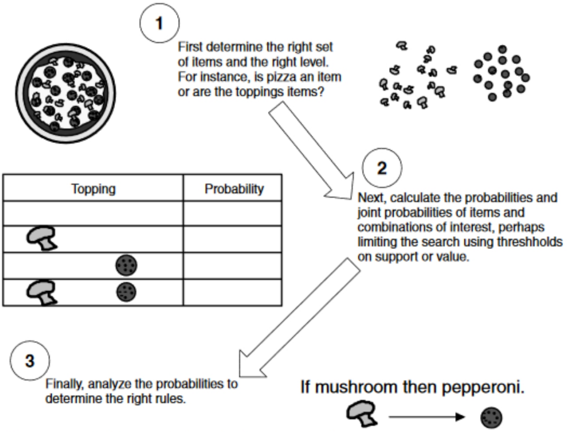

1
First determine the right set of items and the right level. For instance, is pizza an item or are the toppings items?
2
Next, calculate the probabilities and joint probabilities of items and combinations of interest, perhaps limiting the search using threshholds on support or value.
Topping
Probability
3
Finally, analyze the probabilities to determine the right rules.
If mushroom then pepperoni.

Figure 9.9 Finding association rules has these basic steps.

Choose the correct set of products

Usually, the data used for identifying association rules is the accurate transaction data collected at the point of sale. The collection and use of these data is a vital part of the application of market basket analysis, which depends crucially on the products chosen for analysis. What constitutes a specific item depends on the needs of the company. In a grocery store where tens of thousands of items are on the shelf, a frozen pizza might be considered an object for analysis — regardless of its toppings (extra cheese, pepperoni, or mushrooms), its crust (extra thick, whole wheat, or white), or its size. Purchasing a large whole wheat vegetarian pizza, therefore, includes the same "frozen pizza" item as purchasing a single-serving pepperoni with extra cheese. A description of those transactions might look like Table 9.3 at this summarized stage.

Table 9.3 Transactions with More Summarized Items

CUSTOMER	PIZZA	MILK	SUGAR	APPLES	COFFEE
1	✓				
2		✓	✓		
3	✓			✓	✓
4		✓			✓
5	✓		✓	✓	✓

On the other side, the manager of frozen foods or a pizza restaurant chain may be very interested in the unique variations of ordered toppings. He or she may decompose an order of pizza into constituent pieces, as shown in Table 9.4.

Table 9.4 Transactions with More Detailed Items

CUSTOMER	EXTRA CHEESE	ONIONS	PEPPERS	MUSHROOMS	OLIVES
1	✓	✓			✓
2			✓		
3	✓	✓		✓	
4		✓			✓
5	✓		✓	✓	✓

The grocery store may become involved in more detail in its transactions at some later point in time, so that the single item "frozen pizza" would no longer be appropriate. Or the pizza restaurants could extend their choice of menu and become less interested in all the different toppings. Interest things may change over time. This can cause some problem when attempting to use historical data while extracting various levels of information. Choosing the correct degree of detail is key to the study. If the data transaction in the grocery store or any shop keeps track of every form, brand, and size of frozen pizza — which probably accounts for several hundred products — then all these things need to map for analysis to the "frozen pizza" object.

Help to generalize objects /Product Hierarchy.

In the real world, products have product codes and codes for stock-keeping units (SKUs), which fall into hierarchical categories (see Fig. 9.10), called a product hierarchy or taxonomy. Which category hierarchy level is the correct one to use? This brings with it questions like:

- Are big fries and little fries the same product?
- Ice-cream brand is more important than its flavor?
- Which one is more important: the scale, style, pattern, or the clothing designer?

- Is the energy-saving alternative representative of consumer actions on a large appliance?

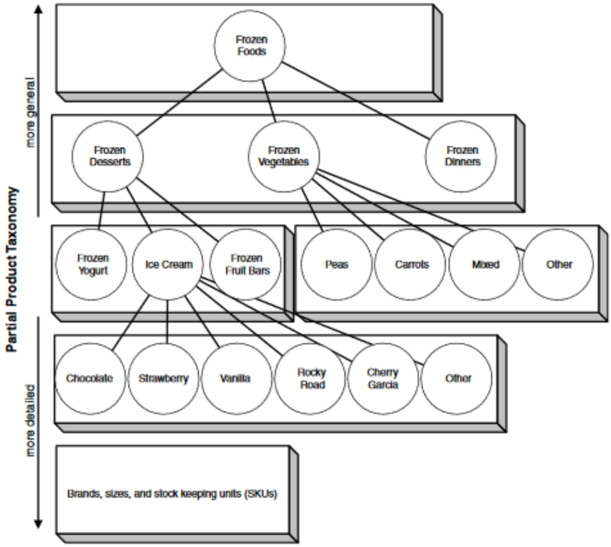

Partial Product Taxonomy
more general
more detailed
Frozen Foods
Frozen Desserts
Frozen Vegetables
Frozen Dinners
Frozen Yogurt
Ice Cream
Frozen Fruit Bars
Peas
Carrots
Mixed
Other
Chocolate
Strawberry
Vanilla
Rocky Road
Cherry Garcia
Other
Brands, sizes, and stock keeping units (SKUs)

Figure 9.10 Product hierarchies start with the most general and move to increasing detail.

If the number of products used in the study increases, the number of variations to consider grows very quickly. This indicates the use of products from higher category hierarchy levels, "frozen sweets" rather than "ice cream." On the other side, the more precise the products are, the more likely the findings would be actionable. Knowing what sells for a specific frozen pizza brand, for example, will aid in controlling the manufacturer's relationship. One solution is first using more general items, then repeating the rule generation to hone in on more unique objects. Since the study focuses on more unique items, just use the subset of transactions that include such items.

The difficulty of law is the number of items that it includes. The longer it takes to create rules of a given complexity, the more products in the transactions. So the required complexity of the rules also defines how things should be specific or general. Customers don't make big transactions under certain conditions. For example, customers buy very few items at a convenience store or through some catalogs at any time, so searching for rules containing four or more items that refer to very few transactions and may be a waste of effort. In other instances, the average transaction is higher, such as in supermarkets, so more complex rules are useful. Moving up, the hierarchy of goods decreases the number of objects.

You may reduce dozens or hundreds of products to a single generic object, often referring to a single department or product line. An object such as a pint of Ben & Jerry's Cherry Garcia becomes generalized to "ice cream" or "frozen food." Instead of investigating "orange juice," investigating "fruit juices," etc. Often the required level of hierarchy ends up matching a department with a product-line manager, so the practical result of identifying interdepartmental relationships is to use categories. Even abstract things help to find laws that are understood sufficiently.

There will be three times as many transactions underpinned by higher taxonomy levels than lower ones. Just because certain items are generalized doesn't mean that you need to move all items to the same level. The acceptable level depends on the object, the emphasis it places on generating actionable results, and the frequency in the data. Big-ticket items (such as appliances) in a department store, for example, may remain at a low level in the hierarchy, whereas less costly items (such as books) may be higher. Often helpful when looking at individual goods is this hybrid approach. Since the data also includes thousands of items, generalize something other than the important product or products.

Virtual items Go beyond the Product Hierarchy

Virtual items are intended to allow the study to take advantage of the knowledge that goes beyond the product hierarchy. In the category hierarchy of the original products, virtual objects do not exist because they cross category boundaries. Examples of virtual goods may include designer labels such as Calvin Klein that appear in both clothing and perfume stores, low-fat and no-fat products in a grocery store, and energy-saving choices on appliances.

Digital products may also contain information about their own purchases, such as whether the payment was made with credit card, cash or check, and the day of the week or the time of the day the transaction took place. Still, crowding the data with too many virtual objects isn't a good idea. If included in well-supported, high-confidence association guidelines, include only virtual objects when you have any understanding of how they will result in actionable knowledge. And there is a risk. Digital objects may cause the rules to be trivial. Imagine, for example, that there is a virtual item for "diet product" and one for "coke product," then a rule will appear as follows: if "coke product" and "diet product" then "diet coke" that is, wherever it appears in a basket and appears in a basket, then it will also appear.

Any basket which has Diet Coke meets this law. While some baskets can have regular coke and other diet items, since it is the concept of Diet Coke, the rule would be a high lift. It is worth testing and rechecking the rules while using virtual objects to ensure that no other trivial rules occur. A related yet subtler risk exists when the corresponding object is not placed on the right-hand side. So, a rule like if "diet product" and "coke product" would probably mean "pretzels," if "diet coke" would mean "pretzels," the only risk of making such rules is that they will confuse what is going on.

9.2 Quality of Data Analysis

In general, the data used for the analysis of market baskets are not of very high quality. It is obtained directly at the point of customer interaction and used primarily for operational purposes such as management of inventories. It is possible that the data will have several formats, corrections, forms of incompatible code, etc. Most of the clarification of different code values are likely to be hidden deep in computer code running in legacy systems and can be hard to retrieve. Often various stores under a single chain have slightly different stock hierarchies or different ways to manage issues such as discounts. Here is only one example.

While in a wide collection of transaction details, the authors were curious about the approximately 80 department codes in there. The client told us of having 40 divisions and offered a good overview of each. More meticulous inspection revealed the problem. Some stores have IBM cash registers, and others have NCR. The two types of equipment's had different ways to interpret department codes — so we saw a lot of invalid data codes. These kinds of problems are characteristic of data mining when using some kind of data.

However, they are intensified for analysis of market baskets since this form of analysis is highly dependent on the un-summarized point-of-sale transactions.

Identified vs. Anonymous

Analysis of consumer baskets has proved useful for mass-market retailers, such as supermarkets, grocery stores, drug stores, and fast-food chains, where many of the transactions have historically been made with cash. Cash purchases are anonymous, meaning the store has no knowledge of particular customers because the transaction does not contain any type information or data that identifies the customers. The only details for anonymous purchases are the date and time, the store's name, the cashier, the items purchased, any coupons redeemed, and the amount of change. Even those minimal data will produce insightful and actionable results with market basket analysis. The proliferation of Online purchases, loyalty schemes, and pure-chasing clubs contributes to increasingly defined purchases, providing analysts with more opportunities for customer information and activity over time.

Data from the consumer basket explains what the consumers buy. Analysis of these data is complicated, and no single method is sufficiently effective to provide all the answers. Usually, the data itself represents the market basket on three separate levels. The order is the buying event; the line-items are the products in the transaction, and the customer transfers orders over time together. Looking at product purchases over time will address several important questions regarding consumer behavior. The things that better sell? What products that sold well last year do not sell well this year anymore? Curves in inventory do not include data on transaction levels. Maybe the most significant insight they offer is the impact of marketing interventions — had sales gone up or down after a specific event? Stock curves, however, are not enough to recognize the relationships between products in a single container. One strategy that is very effective is the rules of the association. This technique will find products that appear to sell in groups together.

Occasionally the groups are enough for perspective. Other times, the classes become clear rules — when some objects are present, then we expect to see some other objects in the container. There are three Association Rules steps. Help says how much transaction data retrieves the law. Trust means how many if the "if" part is real, the "then" part is true, too. And, lift says how much easier to foresee the "then" component than to have no law at all. The rules thus brought about dropping into three groups. Useful rules describe a maybe unforeseen relatedness. Trivial rules describe the relationships known (or ought to be known) to exist. And unexplained laws that don't make much sense. Inexplainable laws also have poor backing.

Analysis of market baskets and association rules offer ways of evaluating item-level information where the relationships between items are decided by the baskets into which they fall. In the next chapter, we will turn to the study of ties, which generalizes the ideas of "objects" connected by "relationships," using the context of a mathematics field called the theory of graph.

Chapter 10: Link Analysis

British Airways and Air France's international route maps provide more than just help with trip planning. They also offer insights into the history and politics of their respective homelands and the empires they lost. A traveler bound from New York to Mombasa at Heathrow is changing planes, one bound at Charles de Gaul for Abidjan changes. The international route maps demonstrate how much information can be gained from understanding the link between items. What other websites link to this? Who is calling who on the phone? What doctors prescribe the pharmaceutical products to which patients? All of these relationships are evident in data, and all of them provide a wealth of information that most data mining techniques cannot take advantage of directly.

In our ever more linked world (where it has been claimed that there are no more than about six degrees of separation between any two individuals on the planet), it is important to consider relationships and linkages. Link analyzes are the techniques of data mining which address this need. Analysis of the relationship is based on a mathematics branch called the graph theory. This chapter discusses the main notions of graphs and then demonstrates how to apply relation analysis to solve real problems. Connection analysis does not extend to all data types, nor does it address all types of problems. However, it also provides very informative and actionable insights when it can be used. Identifying authoritative sources of knowledge on the World Wide Web by examining the connections between its pages are some places where it has produced good results.

Analyzing telephone call patterns to identify unique segments of the market, such as homeworkers.

Comprehension of physician referral patterns; referral is a relationship between two physicians, again, naturally susceptible to linking analysis. Even when links are specifically registered, assembling them into a useful graph can be a data processing challenge.

Links between Web pages are encoded in the pages' HTML itself.

Links between phones are documented in records of the specifics of the call. However, none of these data sources is useful for the study of connections without extensive preprocessing. In other cases, the ties are implicit, and identifying them is part of the data mining challenge. This chapter begins with a small brief introduction to graph theory and some of the classic problems used to solve it. Then it goes on to data mining applications such as search engine rankings and calls information record analysis.

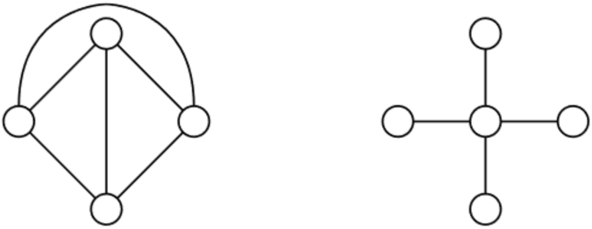
10.1 The Theory of Basic Graphs

Graphs are an abstraction primarily created for the representation of connections. In both mathematics and computer science, they have proven to be very useful in designing algorithms that exploit those relations. Fortunately, graphs are very intuitive, and there are a number of examples to demonstrate how to use them. A graph is composed of two separate parts:

- Nodes (sometimes referred to as vertices) are objects in the graph which have relations. They have names and also have other useful properties.

- Edges are pairs of relationship-connected nodes. The two nodes which it connects represent an edge, so (A, B) or AB represents the edge, which connects A and B.

Even an edge in a weighted graph may have weight. It shows two graphs in Figure 10.1.



A fully connected graph with four nodes and six edges. In a fully connected graph, there is an edge between every pair of nodes.

A graph with five nodes and four edges.

Figure 10.1 Two examples of graphs.

The graph, which is on the left, has four nodes connected by six edges, and it has the characteristic that each pair of nodes has an edge between them. Such a graph is said to be connected to the complete. On an airline where these four cities act as regional hubs, it may reflect regular flights between Atlanta, New York, Cincinnati, and Salt Lake City. It may also represent four people, all of whom know each other, or four leads for a criminal investigation connected to one another. The graph to the right has one node connected to four other nodes in the middle. This may reflect regular flights linking Atlanta to Birmingham and Savannah on an airline serving the Southeast from a hub in Atlanta, or a restaurant that is frequented by four credit card clients. The graph itself gathers the details relevant to what. It can define a great many different conditions without any marks. That is the abstract force. A few terminology points about graphs. Since graphs are so useful to visualize relationships, it is good to draw the nodes and edges without intersecting edges. The charts found in Figure 10.2 have this land.

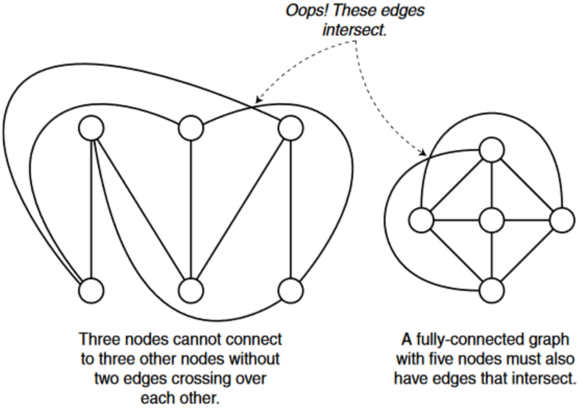

Oops! These edges intersect.
Three nodes cannot connect to three other nodes without two edges crossing over each other.
A fully-connected graph with five nodes must also have edges that intersect.

Figure 10.2 Not all graphs can be drawn without having some edges cross over each other.

They are planar graphs since they can be drawn on a paper sheet (what mathematicians call a plane) without intersecting any edges. Figure 10.2 shows two graphs that cannot be drawn without having a cross of at least two edges. There is, in fact, a theorem in graph theory that says that one of the two graphs previously mentioned is if a graph is nonplanar, then hidden within it. Where there is a way between any two nodes in a graph, it is assumed that the graph is connected. For the remainder, we assume that, unless specified, all graphs are connected. A path, as indicated by its name, is an ordered sequence of nodes connected by edges. Consider a graph where each node in graph represents a city, and the edges are intercity flights. A node is a city on such a graph, and an edge is a section of flight — two cities connected by a non-stop flight. A route is a flight segment itinerary that runs from one city to another, such as from Greenville, South Carolina, Atlanta, Chicago, etc.

Figure 10.3 is an example of a weighted graph, one in which the edges correspond to the weights. In this case, the nodes reflect goods that customers strictly seek. The weights on the edges reflect the association's help, the percentage of market baskets that contain both items. Such graphs provide an approach to solving market basket analysis problems and are also a useful tool for viewing market basket data. This association graph for a product is an example of an undirected graph.

The graph shows that at this organic food store, 22.12 percent of market baskets contain both yellow peppers and bananas. This does not by itself justify whether yellow pepper sales drive banana sales or vice versa, or whether anything else drives buying all yellow fruits and vegetables. A common problem in the study of the ties is to find the shortest path between two nodes. However, it depends on the weights allocated to the edges, which is the shortest. Find the intercity flight charts. Does it apply to the distance for the shortest? To the fewest number of segments in a flight? To flight time as short as possible? Or to the cheapest? Both of certain questions are answered in the same manner using graphs — the only difference is the edge weights. The following two sections describe two classic graph theory problems that illustrate the power of graphs to represent and solve issues. Few problems with data mining are exactly like these two problems, but the problems provide a taste of how simple graph construction leads to some interesting solutions. They are introduced to familiarize the reader with graphs by presenting examples of key concepts in graph theory and presenting a stronger basis for discussion of the study of the connections.

Seven Königsberg Bridges

One of the most advanced problems in graph theory stemmed from a simple challenge presented by Swiss mathematician Leonhard Euler in the eighteenth century. As shown in the simple map in Figure 10.4,

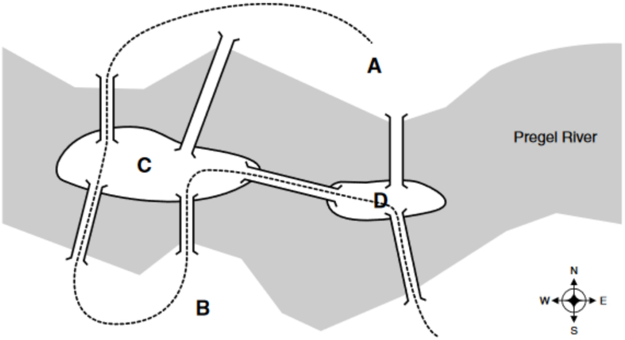


Figure 10.4 The Pregel River in Königsberg has two islands connected by a total of seven bridges.

Königsberg had two islands linked by a total of seven bridges in the Pregel River to each other and to the rest of the area. It is possible to get to either of the bridges on either side of the river or on the islands. Figure 10.4 shows one path through the city, which crosses exactly once over five bridges. Euler asked: Is it possible to walk exactly once across all seven bridges, starting from anywhere in the area, without getting wet or using a boat? As a historical note, the topic lasted longer than the town's name. Königsberg was a famous Prussian city nestled between Lithuania and Poland on the Baltic Sea during the 18th century. Now it's known as Kaliningrad, the westernmost Russian region isolated by Lithuania and Belarus from the rest of Russia. Euler invented graph notation to solve this problem.

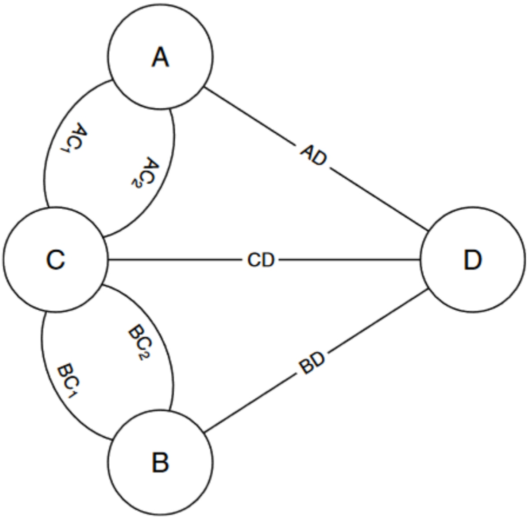


Figure 10.5 This graph represents the layout of Königsberg. The edges are bridges and the nodes are the riverbanks and islands.

In Figure 10.5, he reproduced the Königsberg map as a simple graph with four vertices and seven edges. Some pairs of nodes are bound to each other by more than one edge, suggesting more than one bridge between them. Finding a route that traverses all the Königsberg bridges exactly once is tantamount to finding a path in the graph that reaches each edge exactly once. Such a path is named the Eulerian path in honor of the mathematician who posed this problem and solved it.

Euler has formulated a solution based on the number of edges entering or leaving each node in the graph. The number of such edges is called node degree. For example, in the graph illustrating Königsberg's seven bridges, the nodes representing the shores both have a degree of three — corresponding to the fact that three bridges link each shore to the islands. The other two nodes, which represent the islands, have grades 5 and 3. Euler showed that a Eulerian path only occurs when there are even degrees of all the nodes in a graph, except for at most two (see technical aside). But there's no way to walk across Königsberg's seven bridges without crossing a bridge more than once, as there are four nodes whose degrees are different.

A more recent dilemma in graph theory is the "Traveling Salesman

Problem." In this question, a salesman needs to visit customers in a collection of cities. He wants to travel to one of the cities, rent a car, visit the customer there and then drive to one of the other cities to see one of his customers. In the last place, he leaves the car and flies home. The salesman will take many different routes. Which route minimizes the total distance he travels, thus allowing him to visit every city exactly once? The Traveling Salesman Problem is easily rephrased using graphs, as graphs are a natural representation of road-connected cities.

The edges are cities in the graph representing this problem, and each edge has a weight that corresponds to the distance between the two edge-connected cities. Therefore the Traveling Salesman Problem asks: "What is the shortest path that exactly once reaches all the nodes in a graph? "Note that this issue is different from Königsberg's Seven Bridges. We are not interested in seeking a path simultaneously that reaches all nodes exactly once, but we want the shortest of all possible paths. Note that all Eulerian paths have exactly the same length since they have the same edges. It is not sensible to request for the shortest Eulerian course. It is not difficult to solve the Traveling Salesman problem for three or four cities. The most complex graph with four nodes is a fully connected graph, in which each node in the graph is linked to each other. In this graph, 24 separate paths exactly once visit each node. Start at any of the nodes to count the number of paths (there are four possibilities), then go to each of the remaining three, then to each of the other two, and ultimately to the last node (4 * 3 * 2 * 1 = 4! = 24). A fully connected graph containing n nodes has n! (N factorial) independent paths, comprising all nodes.

Each path has a slightly differentiated edge set, so their lengths are typically different. Since listing the 24 possible paths isn't that difficult for this specific case, finding the shortest path is not especially difficult. The Irish mathematician Sir William Rowan Hamilton first invested the problem of finding the shortest path connecting nodes. Study of minimizing energy in physical or real systems led him to explore energy minimization in some discrete systems that he described as graphs.

In his memory, a path that exactly once visits all nodes in a graph is called a Hamiltonian path. The problem with the Traveling

Salesman is hard to fix. Both possible paths through the graph must be considered by any solution to decide which one is the shortest. The number of paths in a fully connected graph is increasing very rapidly — as a factorial. What is true for fully connected graphs is usually true for graphs: the number of possible paths that all nodes follow grows like an exponential function of nodes (although there are a few simple graphs where this is not valid). So as the number of towns increases, the effort needed to find the shortest route increases exponentially. Adding only one more town (with related roads) will lead to a solution that takes twice as long — or more — to find.

This lack of scalability is so significant that mathematicians gave it a name: NP — where NP means all known algorithms were used exponentially to solve the problem scale — not like a polynomial. These questions are difficult to understand. The Traveling Salesman Dilemma is often so complex that it is used to test parallel computers and unconventional computational methods — such as using DNA or the complexities of quantum physics as the source of computers instead of the more common computer chips made from silicon.

Aside from all this graph theory, there are pretty good heuristic algorithms for computers that provide rational solutions to the problem of traveling salesmen. The resulting paths are relatively short or small paths, although the shortest possible one is not guaranteed to be as short as possible. If you have a similar problem, this is a useful fact. One popular algorithm is the greedy algorithm: start the path with the graph's shortest edge, then lengthen the path with the shortest edge at either end, visiting a new node available. In general, the resulting path is very short, but not necessarily the shortest (see Figure 10.6).

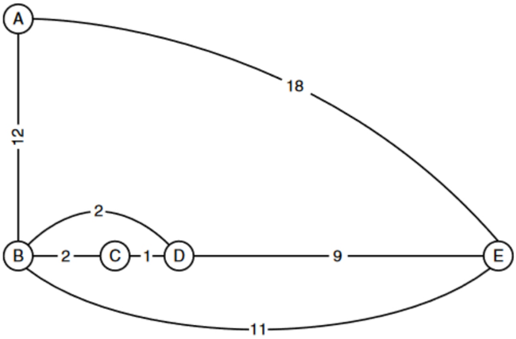


Figure 10.6 In this graph, the shortest path (ABCDE) has a length of 24, but the greedy algorithm finds a much longer path (CDBEA).

The graphs discussed until now are undirected. In undirected graphs, the edges between nodes are like expressways: they go in both directions. The edges are like one-way paths in a directed graph. An edge that goes from A to B is distinct from a boundary that goes from B to A. A guided edge from A to B is an outgoing edge of A, and an entrance edge of B. Driven graphs reflect the data in a powerful way:

- Flight segments linking a collection of cities
- Hyperlinks between Websites
- The calling habits by phone
- Two forms of nodes are of particular interest in the form of graphs.

All edges linked to a source node are edges that are outgoing. Since there are no incoming edges, there is no path to any of the source nodes from any other node in the graph. When all the edges are incoming edges on a node, the node is called a sink node. The presence of source nodes and sink nodes is a major difference between them and their undirected relatives.

Whether the graph comprises any paths that start and end at the same vertex is an essential property of the directed graphs. Such a path is called a loop, meaning that the path could repeatedly repeat itself: ABCABCABC, etc. If at least one cycle is present in a directed graph, it is called cyclic. For example, cycles in a flight segment graph may be the path of a single plane. Members of a loop call each other in a call graph — these are good candidates for a "friends and family-style", where the whole party gets a discount or selling conference call services. Detection of Cycles in a Graph There is a simple algorithm to detect if there are cycles in a directed graph.

This algorithm begins with the observation that if a directed graph has no sink vertices and at least one vertex, then any direction can be arbitrarily extended. The terminating node of a specific path is always connected to another node without any sink vertices so that the path can be expanded by appending that node. Similarly, if there are no source nodes in the graph, then we can always prepend a node to the path start. If the path includes more nodes than the nodes in the graph, we know that at least one node has to be visited twice. This node is called X. The portion of the path in the path between the first X and the second X is a circle, so the map is cyclic. Now consider the case where there are one or more source nodes in a graph and one or more sink nodes.

The fact that source nodes and sink nodes cannot be part of a loop is very clear. Removing the source nodes from the graph together with all its edges does not change the cyclic structure of the graph. If the resulting graph has almost no sink nodes or no source nodes, then as just seen, it contains a loop.

Repeat eliminating sink nodes, source nodes, and their edges before one of the following happen:

No more edges are left or no more nodes. In this case, there are no cycles to the graph.

While some edges remain, no source or sink nodes exist. The diagram, in this case, is cyclic. If there are no loops, then the graph is considered an acyclic graph.

These graphs are useful to explain the dependencies or relationships between items in one way. Different products, for example, sometimes belong to nested hierarchies that can be described by acyclic graphs. A further example is the decision trees mentioned in Chapter 6. Any two nodes in an acyclic graph have a well-defined relationship of precedence with each other. If in any path that contains both A and B, node A precedes node B, then A precedes B in all paths that contain both A and B (otherwise, a loop would be). We are saying, in this case, that A is a predecessor of B, and that B is a successor of A. If there are no paths that contain both A and B, then A and B are disjointed. This strict ordering can be an essential node property and is often useful for the purposes of data mining.

10.2 A Family Relation Research Framework

Many of this book's readers have possibly used the search engine Google. Her phenomenal popularity stems from her ability to help people find reasonably good material on almost any subject. This feat is achieved by examination of the ties. The World Wide Web is a huge graph that is driven. The nodes are Web pages and that edges between them are hyperlinks. Special programs called spiders or web crawlers traverse these connections on an ongoing basis to update maps of the vast directed graph that is the network. Some of these spiders actually index the content of Web pages to be used by search engines based on pure text. Others document the global structure of the Network as a directional graph which can be used for analysis. Once upon a time, search engines only evaluated the graph's nodes. Google's strategy (which other search engines have now adopted) is to make use of the information stored in the edges of the graph as well as the information contained in the nodes.

Some blogs or magazine articles are more interesting than others, even though they are dedicated to the same subject. This basic concept is much easier to understand but difficult to convey to a machine. So when a search is conducted on a subject that many people write about, finding the most important or reliable documents in the vast set that fits the search criteria is difficult to find. Cornell University Professor Jon Kleinberg has come up with one widely adopted technique to address this problem. His methodology takes advantage of the intuition that a human being makes a decision about the importance of the site to which he is connected while making a connection from one site to another.

Any connection to another site is actually a recommendation from that site. Cumulatively, the independent decisions of multiple designers of websites that all intend to provide links to the same goal grant authority on that aim. In addition, the reliability of the linked sites may be measured on the basis of the authority of the sites to which they relate. A site's recommendations with tons of other positive recommendations can be given more weight in deciding another's authority. A page that possibly links to many authorities is a hub in Kleinberg's terminology; authority is a page that is linked to almost every hubs. The two definitions can be used together to say the difference between authority and mere popularity, as shown in Figure 10.7.

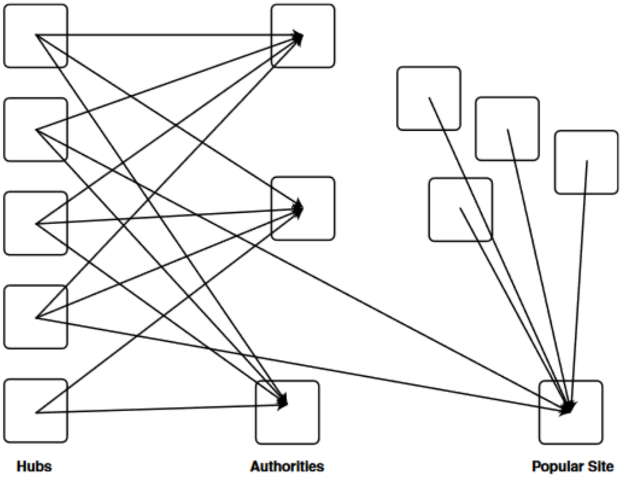


Figure 10.7 Google uses link analysis to distinguish hubs, authorities, and popular pages.

At first glance, a good way to identify authoritative websites would seem to be to list them by the number of unrelated websites that connect to them. The problem with this strategy is that if a common site (one with several inbound links) mentions the subject, even in passing, it will be ranked higher than a site that is much more authoritative on the specific topic, but less common overall. The solution is to rate pages not by the total number of links pointing to them, but by the number of hubs pointing to them in relation to the topic. Google.com makes use of an updated and improved version of the Kleinberg basic algorithm mentioned here. A quest-based on an interpretation of the links starts with an ordinary quest based on text.

This initial search provides a pool of pages (often a few hundred) to start the process with. The collection of documents returned by such a search is very likely not to contain the documents that a human reader would find to be the most authoritative source on the subject. This is because the most authoritative sources on a subject are not always the ones who use the terms most often in the search string. Kleinberg is using the example of a "Harvard" keyword search. Most people will consent to this www. Harvard.edu is one of the most authoritative sites on this subject, but, in a strictly content-based review, it does not stand out from the more than about 1 million web pages containing the word "Harvard," so it is very likely that a text-based search would not return the university's own website among the top results. However, it is highly possible that at least a few of the returned documents will include a link to Harvard's home page or, failing that, that any page referring to one of the pages in the pages pool would also point to www.harvard.edu. An important feature of Kleinberg's algorithm is that it doesn't just take the pages returned by the initial text-based search and attempt to rank them; it uses them to create a much larger pool of documents pointing to or pointing to by any of the root-set documents. This larger pool includes a much more global structure — structure that can be mined to decide which documents the vast group of people who created the documents in the pool is considered to be most authoritative of.

Analyzing the relation is an application of the graph theory mathematical area. Link analysis as a data mining technique has many strengths:

- The partnership capitalizes.
- The visualization is useful.
- It produces derivative features that can be used for further mining.

Some issues with data mining and data mining inevitably include ties.

As the two case studies on telephone data show, the study of connections is very useful in telecommunications — a telephone call is a connection between two individuals. In fields where the similarities are obvious, such as telephony, transportation, and the World Wide Web, possibilities for linking research are most evident. Relation analysis is also necessary for other fields where there is not such a simple representation of the links, such as physician referral trends, retail sales data, and criminal forensic analysis.

Links are a very common way of visualizing such data forms. Simple visualization of the connections can be a great help for the exploration of information. And when auto-mated patterns are detected, connection visualization helps to understand better what's going on. Link analysis provides an alternative way to look at the data, different from relational database formats and OLAP tools. Links may indicate important patterns in the data, but an individual is needed to interpret the significance of the patterns. Analysis of the links will lead to new, useful data attributes. Examples include calculating a score of authority for a World Wide Web website and measuring the sphere of control for a telephone user.

While, where applicable, connection analysis is very efficient, it is not suitable for all types of problems. It is not a predictive tool or classification tool such as a neural network that takes in data and generates a response. Many data types are simply not suitable for the study of the connections. Its best use is likely to be in identifying particular trends, such as outgoing call forms, which can then be applied to data. For use in combination with other guided data mining techniques, these patterns can be converted into new data features.

Chapter 11: Automatic Clusters Detection In Data Mining

To find meaningful patterns in data, the data mining techniques described in this book are used. These patterns aren't always coming straight away. This is because there are no patterns or models to be found. Other times, the problem is not pattern shortages but excess. The data can contain so much complex structure that even the best techniques of data mining cannot coax meaningful patterns out. When mining a database like this to answer some specific questions, competing explanations tend to cancel out each other. Too many competing signals add up to the noise as with radio reception.

Clustering provides a process in the way to learn about the complex data structure, breaking up the cacophony of competing signals into its parts or components. When humans try to give meaning to complex questions, our natural or common tendency is to break the subject into small pieces, each of which can be explained more simply. If someone were asked to describe the forest's color of trees, the answer would probably distinguish between deciduous trees, and between winter, spring, summer, and fall. People may know enough about woodland flora to predict that the best factors to use to form clusters of trees that follow similar coloring rules are the hundreds of variables associated with the forest, season, and foliage class, rather than say height and age. Simple patterns can often be identified within each cluster after the proper clusters have been established. "Deciduous trees don't have leaves in winter, so the trees tend to be brown" or "Deciduous trees leaves change color in fall, usually to oranges, reds, and yellows." In many situations, a very noisy dataset simply consists of a number of better-behaved clusters.

The question is: how can these find themselves? This is where cluster automatic detection strategies come in — to help see the forest without being lost in the woods. This chapter starts with two examples of clustering's usefulness — one is taken from astronomy, one from clothing design. It then introduces the algorithm of K-Means clustering , which depends on a geometric interpretation of data, like the nearest neighboring techniques discussed in Chapter 8.

The geo-metric ideas used in K-Means raise the more general question of similarity, relation, and distance measurements. These distance measurements are very sensitive to variations in the representation of data, so the next subject discussed is clustering data planning, with particular attention being paid to scaling and weighting. K-Means is not the most widely used algorithm for the automated detection of clusters. This chapter includes several other brief discussions: Gaussian mixture models, agglomeration clustering, and divisive clustering. (Another clustering technique, self-organizing maps, is also covered in Chapter 7 since self-organizing maps are a type of neural network.) The chapter concludes in the end with a case study in which automatic cluster detection is used to test editorial zones for a major daily newspaper.

Searching for Simplicity Islands

Data mining techniques are classified as direct or undirected, and automatic cluster detection is defined as a tool for the discovery of undirected information. That is valid in the technical sense since the automated cluster detection algorithms themselves simply find a structure that occurs in the data without regard to any unique target variable. Most data mining tasks begin with a pre-classified training set that is used to create a model that can score or classify previously unseen information. There is no pre-classified data in clustering, nor is there any distinction between independent and dependent variables. Instead, clustering algorithms look for groups of records — the clusters — composed of related records. These similarities are discovered by algorithms. It is up to the people who run the study to decide whether similar documents are of interest to the business — or something mysterious and maybe unimportant. However, clustering may be a focused practice in a wider context since clusters are pursued for some business reasons. Clusters developed for a business purpose are generally called "segments" in marketing, and clustering is a common method of customer segmentation. Automatic cluster detection is a technique of data mining that is rarely used in isolation because cluster discovery is not always an end in itself. When clusters are observed, other approaches must be utilized to decide what the clusters represent.

The results can be dramatic when clustering is successful: One famous early application of cluster detection led to our current underlying stellar evolution.

Light Of The Moon, Bright Star

Astronomers attempting to explain the relationship between the luminosity (brightness) of stars and their temperatures in the early twentieth century rendered scatter plots like the one in Figure 11.1. The vertical scale measures luminosity in multiples of our own sun's light. The horizontal scale calculates surface temperature in degrees Kelvin (degrees centigrade above absolute 0, the coldest theoretical temperature possible). Two different astronomers, Denmark's Enjar Hertzsprung and US-based Norris Russell thought about doing this at around the same time. They both observed the stars dropping into three clusters in the resulting scatter map. This discovery led to further study and an understanding that in very different phases of the stellar life cycle, these three clusters reflect stars.

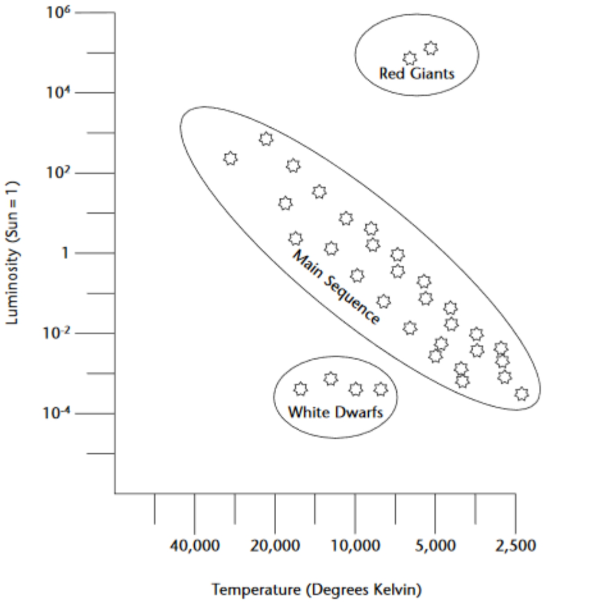


Figure 11.1 The Hertzsprung-Russell diagram clusters stars by temperature and luminosity.

The relationship between luminosity and temperature within each cluster is similar, but the relationship between the clusters is different since the heat and light are produced by fundamentally different processes. By converting hydrogen to helium by nuclear fusion, 80 percent of stars that fall on the main sequence produce energy. This is how the majority of all-stars spend their productive lives. The hydrogen is used up after several billions of years. The star either starts to fuse helium, or the fusion ceases, depending on its mass. In the latter case, the star's core collapses, producing much heat in the process. At the same time, the outer gaseous layer extends away from the core, forming a red giant. The outer layer of gasses is gradually stripped away, and the remaining core begins to cool off. Today the star is a white dwarf.

A recent Google search using the word "Hertzsprung-Russell Diagram" returned thousands of pages of links to current astronomical research focused on this sort of cluster detection. Even today, HR-based clusters are used to search brown dwarfs (strong objects lacking sufficient mass to induce nuclear fusion) and to consider pre-main sequence of stellar evolution.

A diagram of Fitting the Troops The Hertzsprung-Russell is a liable introductory example of clustering because it is simple to identify the clusters visually with only two variables (and, interestingly, it is a good example of the value of good visualizations of data). Also, in three dimensions, it is not too difficult to pick out clusters from a scatter plot cube by eye. If all of the problems had too few dimensions, automatic cluster detection algorithms would not be necessary. As the quantity of dimensions (independent variables) increases, cluster visualization becomes progressively difficult. Even our understanding of how close objects are to each other breaks down easily with more dimensions. To claim that there are many dimensions to a problem is an invitation to explore it geometrically.

Both of the items that must be calculated separately in order to define something are one dimension. In other words, if there are N variables, imagine a space where each variable's value represents a distance in an N-dimensional space along the corresponding axis. For each of the N variables, a single record containing a value can be viewed as the vector that defines a particular point in that space. That is easily plotted because there are two dimensions. One such example was the HR-diagram. Figure 11.2 is another example of how a group of adolescents maps their height and weight as points on a line. Note how boys and girls mix.

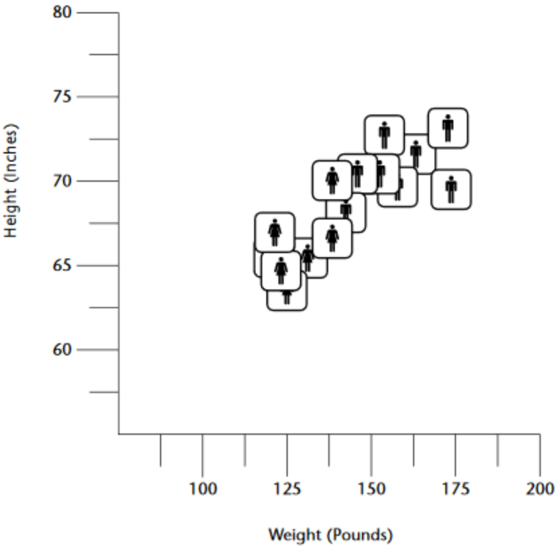


Figure 11.2 Heights and weights of a group of teenagers.

The map in Figure 11.2 starts providing a rough picture of the shapes of the people. But if the objective is to suit them for clothing, we need a few more measurements! In 1990s, the U.S. military conducted a report about how to reshape female soldiers' uniforms. The army's aim was to reduce the amount of different height (Inches) uniform sizes that must be kept in stock, while also supplying well-fitting uniforms to each soldier. As everyone who has ever shopped for women's apparel knows, there is already a surface of classification structures (including measurements, odd measurements, plus sizes, junior, petite, etc.) to categorize apparel by gender. None of these systems have been developed with US military needs in mind. Researchers at Cornell University, Susan Ashdown and Beatrix Paal, went back to basics; they designed a new set of sizes based on the real shapes of women in the military.

Unlike conventional clothing size structures, Ashdown and Paal did not come up with an organized set of graduated sizes where all sizes expand together. Rather, they have come up with sizes that suit various types of bodies. Each type of body corresponds to a cluster of records in a body measurement database. One cluster may consist of short-legged, small-waisted, big-busted women with long torsos, medium bodies, wide shoulders, and thin necks while other clusters capture other measurement constellations—the database comprised over 100 measurements for almost 3,000 women each. The technique used for clustering was the K-means algorithm, defined in the next section. In the last, it took only a few of the more than 100 measurements to classify the clusters. A further advantage of the clustering method was discovering this reduced number of variables.

11.1 K-Means Clustering

One of the most widely and extensively clustering algorithms used is the K-means algorithm. The "K" in its name relates to the fact that the algorithm looks for a fixed number of clusters that are identified in terms of data point proximity to one another. The version listed here was published first by J. 1967 B. MacQueen. The technique is demonstrated using two-dimensional diagrams to provide ease of clarification. Note the algorithm typically manages far more than two independent variables in operation. This implies that, instead of points corresponding to vectors with two-elements (x1,x2), the points correspond to vectors with n-elements (x1,x2,. In itself, the process is unchanged.

Three Steps of the K-Means algorithm

The algorithm randomly selects K data points to be the seeds in the first step. MacQueen's algorithm simply takes records of the first K. In cases where the records have some significant order, selecting widely spaced records or a random collection of records might be preferable. -- seed is an embryonic cluster with one element in it. This example sets Cluster Number to 3. The latter step assigns each record to the nearest seed. One way to do that is to find the boundaries between the clusters, as geometrically shown in Figure 11.3. The boundaries between the two clusters are the points on which each cluster is equally close.

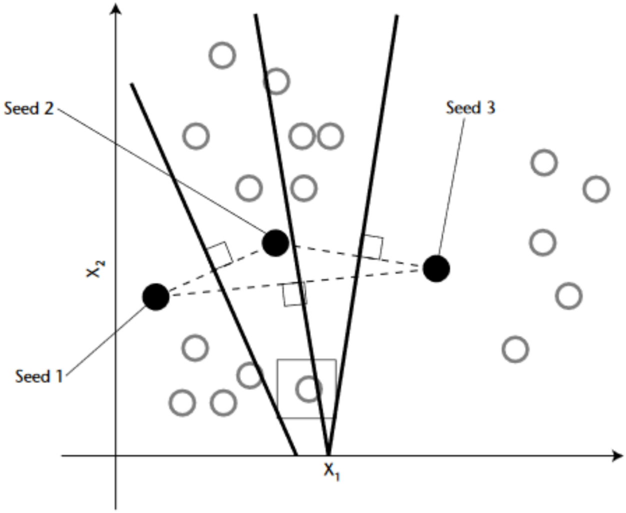


Figure 11.3 The initial seeds determine the initial cluster boundaries.

Recalling a high school geometry lesson makes this less complicated than it sounds: given about any two points i.e., A and B, all points that are almost equidistant from point A and B fall along a line (which is called the perpetual bisector) that is perpendicular to the one that connects A and B and between them. The dashed lines link the initial seeds in Figure 11.3; the resulting cluster boundaries shown with solid lines are in right angles to the dashed lines. Using these lines as guides, it is clear which records are closest to the seeds.

These boundaries would be planes in three dimensions, and they would be N-dimensional hyperplanes N – 1. Fortunately, such conditions are managed easily by computer algorithms. Finding the actual boundaries among the clusters is useful for geometrically displaying the operation. However, in practice, the algorithm normally calculates the distance between each record and each seed and chooses the minimum distance for this step.

Consider, for example, the record with the box drawn around it. Based on the initial seeds, this record is assigned to the seed number 2 managed set because it is likely closer to that seed than to one of the other two. Every point has been allocated at this point to exactly one of the three clusters based around the original seed. The third step is to measure the cluster centroids; these now do a better job of characterizing the clusters than the initial seeds. Finding the centroids is simply about taking the average value of each dimension for all the cluster records. The new centroids are stamped with a cross in Figure 11.4. The arrows display the movement from the location of the original seeds to the cluster new centroids produced from those seeds.

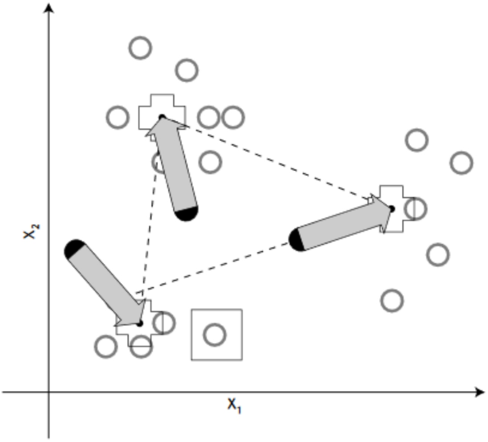


Figure 11.4 The centroids are calculated from the points that are assigned to each cluster.

The centroids become the seeds to the algorithm's next iteration. Step 2 is repeated, and each point with the closest centroid is again allocated to the cluster. Figure 11.5 shows the new boundaries of clusters — formed, as before, by drawing equidistant lines between each pair of centroids.

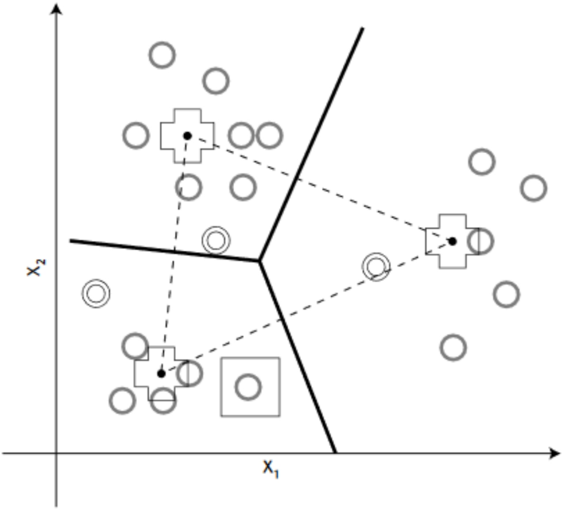


Figure 11.5 At each iteration, all cluster assignments are reevaluated.

Note that cluster number 1 has now been allocated to the point with the box around it, that was originally assigned to cluster number 2, The process of assigning cluster points, and then recalculating centroids, continues until the boundaries of the cluster stop shifting. In practice, after a few tens of iterations, the K-means algorithm typically finds a set of stable clusters.

What does K Mean?

Clusters define the structure that underlies the data. There is no one proper definition of the structure, however. For example, someone who isn't from New York City may think the entire city is "downtown." Someone from Brooklyn or Queens may apply this nomenclature to Manhattan. It could just be areas within Manhattan south of 23rd Street. And even there, in the southern tip of the island, "downtown" could still be reserved only for the taller buildings. Clustering poses a similar problem; data structures occur on several different levels.

K-means definitions and allied algorithms gloss over K's variety. But since there is no a priori justification to choose a specific value in certain situations, there really is an outermost loop to these algorithms that exist in the course of analysis rather than in the computer program. This outer loop consists of automatic cluster detection using one K value, evaluating the results, then retrying with another K value, or perhaps modifying the data. The intensity of the resulting clusters can be evaluated after each trial by comparing the average distance between records in a cluster with the average distance between clusters, and other procedures mentioned in this chapter later. These tests can be automated, but it is often important to evaluate the clusters on a more subjective basis to assess their utility for a given application. As shown in Figure 11.6, different K values can lead to equally valid clusterings of very different types. The figure shows the clustering of a deck of K = 2 and K = 4 playing cards. Is one better than the next? It depends on the use the clusters are put to.

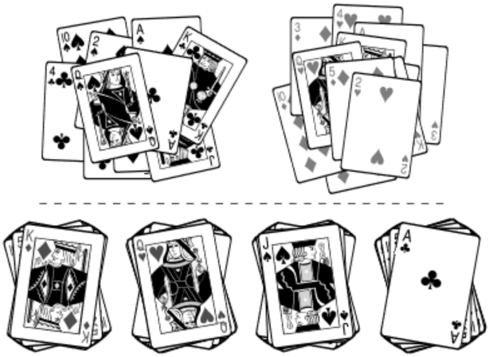


Figure 11.6 These examples of clusters of size 2 and 4 in a deck of playing cards illustrate that there is no one correct clustering.

For the first time, K-means clustering is performed on a given data set, most data points fall into one massive central cluster, and a number of smaller clusters are outside. This is mostly because most records describe "natural" variations in the data, but the clustering algorithm has enough outliers to confuse. This form of clustering can be useful for applications such as fraud detection or fabrication defects. In other cases, removing outliers from the data can be desirable; more often, massaging the data values is the solution.

Distance and Similarity

If records are mapped to space points in a database, identification of automatic clusters is really very simple — a little geometry, some vector means, and voilà! Of course, the issue is that the systems encountered in marketing, distribution, and customer service are not about space points. They are about sales, phone calls, airplane journeys, car registration, and a thousand other items that have no apparent relation to the dots in a cluster diagram. This kind of clustering records involves some notion of natural association; that is, records in a cluster are equal or related to each other than records in another cluster. Since abstract concepts are hard to express to a machine, his vague definition of connection has needs to be converted into a kind of numerical measure of the degree of similarity. The most popular but by no means the only approach is to convert all fields into numeric values so that the records can be viewed as space points. In the geometric sense, then, if two points are identical, they represent similar records in the database. This method has two key problems: Many variable forms, including all categorical variables and many numerical variables like rankings, do not have the correct behavior to be handled properly as components of a position vector. The contributions dimension is of equal significance in geometry, but in databases, a minor change in one field can be much more significant than a big change in another. A few alternative similarity measures are introduced in the following section.

Similarity Measures and Geometric distance of variable type works well as a similarity measure for well-behaved numeric variables. A numeric variable is one whose value in our geometric model indicates its location along the axis, which corresponds to it. Not all of the variables fall into that group. Variables fall into four classes to this end, listed here in increasing order of suitability for the geometric model.

--Differential variables
--The Ranks
-- Intervals
-- Right measures

Categorical variables define only one of many unordered categories to which one element belongs.

For example, one ice cream pistachio and another butter pecan may be labelled, but it is not possible to tell that one is larger than the other, or to determine which one is closer to black cherry.

Mathematically, $X > Y$ can be assumed, but not if $X > Y$ or $X < Y$. Ranks put things in order, but don't assume that one thing is any bigger than another. The valedictorian has higher grades than the salutatorian but by how much we don't know. If X, Y , and Z are ranked A, B , and C, we know that $X > Y > Z$, but $X-Y$ or $Y-Z$ cannot be specified. Intervals measure the interval from one measurement to another. If it is 56 ° in San Francisco and 78 ° in San Jose then at one end of the bay it is 22 degrees colder than at the other.

True measures are variables of intervals that measure from a meaningful point zero. This trait is important because it means the ratio of the variable's two values is large. The temperature scale of Fahrenheit used in the United States and the scale of Celsius used in the rest of the world have this property. It doesn't make sense in either system to say a 30 ° day is twice as warm as a 15 ° day. Similarly, a size 12 dress isn't twice as big as a size 6, and gypsum isn't twice as hard as talc, but on the hardness scale, they are 2 and 1. But it makes complete sense to say a 50-year-old is twice as old as a 25-year-old or a 10-pound sugar bag is twice as heavy as a 5-pound one.

Examples of true measurements are age, weight, length, customer tenure, and duration. Geometric distance metrics for interval variables and true measurements are well defined. It is important to transform these into interval variables in order to use categorical variables and rankings. Unfortunately, it may apply false details to those transformations. If random numbers 1 through 28 are assigned to ice cream flavors, it will resemble that flavors 5 and 6 are nearly related while aromas 1 and 28 are far apart.

Other Cluster Identification Methods

There are several modifications to the base K-means algorithm. Many commercial software tools, which include automatic detection of clusters, incorporate some of these variations. Alternate methods of selecting the initial seeds and using probability density rather than distance to associate records with clusters are among the differences. This latter variant needs further consideration. Additionally, there are several different clustering methods, including agglomerative clustering, competitive clustering, and self-organizing maps.

11.2 Gaussian Model

As mentioned the K-means method has some drawbacks:

— Doesn't do well with cluster overlaps.

— The outliers quickly drag the clusters off centre.

— Each record is either within or outside of a given cluster.

Models of Gaussian mixture are a probabilistic version of K-means. The name derives from the Gaussian distribution, distribution of probabilities frequently presumed for problems of large dimensions. The Gaussian distribution generalizes to more than one variable the normal distribution. As before, they were selecting K seeds to start with the algorithm. However, this time the seeds are considered the source of Gaussian distributions. The algorithm goes by iterating over two steps, called the step of estimation and the step of maximization. The estimation stage calculates the liability for each data point kept by each Gaussian (see Figure 11.8).

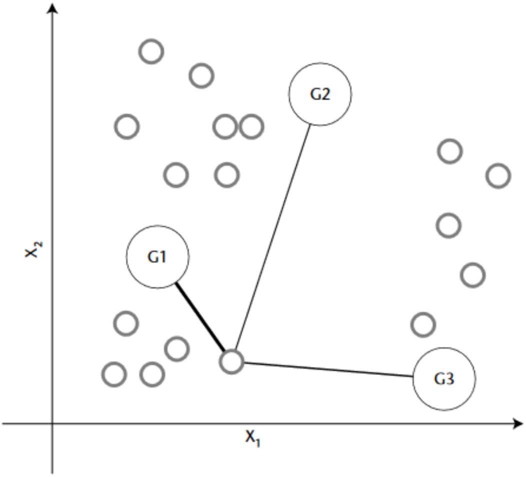


Figure 11.8 In the estimation step, each Gaussian is assigned some responsibility for each point. Thicker lines indicate greater responsibility.

Every Gaussian has a strong responsibility for points close to their mean and a weak responsibility for distant points. In the next step, the roles will be used as weights. A new centroid for each cluster is determined in the maximization stage, taking into account the newly calculated responsibilities. The centroid for a given Gaussian, as shown in Figure 11.9, is determined by averaging all the points weighted for that Gaussian by the answerability.

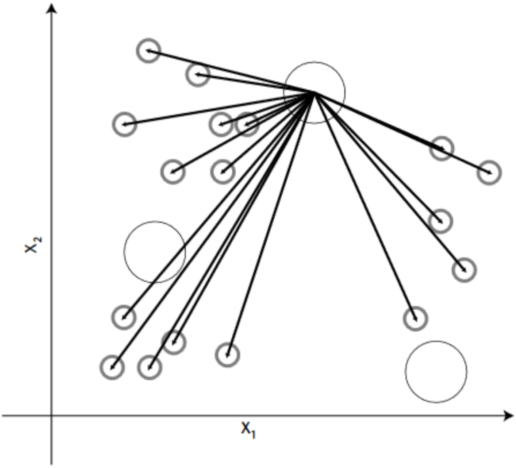


Figure 11.9 Each Gaussian mean is moved to the centroid of all the data points weighted by its responsibilities for each point. Thicker arrows indicate higher weights.

These steps are repeated until the Gaussians do not move anymore. The Gaussians themselves can both change form and travel. Every Gaussian is constrained, however, so if it shows a very high responsibility for points close to its average, then a sharp drop in responsibilities. If the Gaussian encompasses a broader spectrum of values, then he has smaller obligations for points nearby. Given that the distribution must always be integrated into one, Gaussians are always getting weaker as they grow larger. The reason this is called a "mixture model" is because the likelihood is the sum of a mixture of many distributions at each data point. Every point at the end of the process is bound to the different clusters with higher or lower probability. This is often called soft clustering since a single cluster does not classify points uniquely. One indication of this approach is that certain points in more than one cluster can have a high probability. Other points may only have very low odds of being in any cluster. Each point can be assigned to the cluster where it has the highest probability, making this soft clustering a hard clustering.

Automatic cluster detection is an undirected data mining tool, which can be used to learn about complex database structures. By splitting com- plex datasets into simpler clusters, it is possible to use automated clustering to boost the efficiency of more focused techniques.

The automatic clustering can be applied to almost any form of data by choosing different distance measurements. Clusters can be found just as easily in news story collections or insurance reports as in astronomical or financial details. Clustering algorithms can rely on some form of similarity metric to determine whether two records are near or distant. A geometric representation of distance is sometimes used, but there are other possibilities, some of which are more acceptable when non-numeric data is found in the records to be clustered. One of the most common Cluster automatic detection algorithms is K-means.

The K-means algorithm is an iterative approach to remote-based identification of K clusters. Several other algorithms were also included in the chapter. Gaussian mixture models is a variation on the concept of K-means that allows clusters to overlap. Divisive clustering creates a tree of clusters by splitting an initial large cluster in succession. The agglomerative clustering begins with several small clusters and combines them progressively until there is just one cluster remaining. Divisive and agglomerative approaches allow the data miner to use external parameters to assess which cluster tree level is most useful for a specific application. This chapter presented several technical metrics for cluster fitness, but the most critical metric for clustering is how effective the clusters are in achieving a certain business objective.

Summary

Data Mining is an important part of customer relationship management analytics. The purpose of analytic customer relationship management is to replicate the friendly, learning relationship that a well-run small business enjoys with its customers to the extent possible. The interactions within a business and its customers produce large quantities of data. Initially, this data is collected in transaction processing systems such as automated teller machines, phone transfer logs, and files for supermarket scanners. The data can then be compiled, cleaned, and summarized for use in a data warehouse for customers. A well-designed data center for consumers provides a comprehensive record of customer experiences that becomes the corporate memory. Data mining tools can be applied to this historical record to learn customer stuff, which will allow the business to better support them in the future. The chapter presents several examples of commercial data mining applications such as better targeted couponing, making suggestions, cross-selling, customer retention, and reducing credit risk. Data mining itself is the method of discovering useful patterns and rules in large data volumes. This chapter introduced and described six common tasks of data mining: classification, estimation, prediction, a grouping of affinities, clustering, and profiling. The rest of the book explores a number of algorithms and techniques for data mining that can be applied to those six tasks. To thrive, these strategies must become part and parcel of a larger business method.

Data mining comes in two distinct ways. Driven data mining involves researching patterns that explain a particular outcome through historical records. Driven data mining involves classification, prediction, forecasting, and profiling activities. Undirected data mining searches for interesting patterns from the same documents. It involves the clustering tasks, finding the rules of association, and explanation. Data mining gets data closer to the sector. Hypothesis testing, therefore, is a very necessary part of the process. The primary lesson of this chapter, however, is that data mining is full of traps for the unwary, and it can only avoid them by following an experience-based approach.

In support of both ads and direct marketing, data mining is used to define the correct target, select the best messaging platforms, and choose the most relevant messages. Prospective consumers may be matched with the intended public's profile and given a fitness ranking. If there are no details on individual prospects, Data Mining Applications can be used to allocate fitness scores to local communities using available type data from the U.S. census office, Statistics Canada, and similar official sources in several countries.

Methods for decision-tree have broad applicability for data discovery, classification, and ranking. They can also be used to estimate continuous values, but they are rarely the first option because "lumpy" estimates are created by decision trees — all records that hit the same leaf are assigned the same estimated value. They are a good choice when the task of data mining is to identify information or to predict discrete results—using decision trees to assign each record to one of a few large categories when your goal is to. Theoretically, decision trees can assign records to an arbitrary number of classes, but when the number of training examples per class becomes limited, they are prone to error. In a tree with many levels and/or several branches per node, this can happen very quickly. In certain business cases, challenges usually overcome a binary classification such as respondent/nonrespondent or good/bad, so in practice, this is not a major problem.

Neural networks are a flexible method for extracting data. Neural networks have proved themselves time and again through a large range of sectors and a large number of applications. These findings come in complex areas, such as time series analyses and fraud detection, which are not easily adapted to other techniques. The largest neural network designed for output is possibly the device developed by AT&T to read on-check numbers. This neural network consists of hundreds of thousands of units arranged in seven layers.